Prototipo de software para el pronóstico de demanda de energía en la Electrificadora de Santander.

María Fernanda Rodriguez Pulido

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director

Laura Viviana Galvis Carreño

Ph.D. en Ingeniería Eléctrica y Computación.

Escuela de Ingeniería de Sistemas e Informática.

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería de Sistemas e Informática

Ingeniería de Sistemas

Bucaramanga

2024

Dedicatoria

Este proyecto de grado está dedicado:

A Dios, primero y por, sobre todo, por estar tan presente en mi vida y mi familia, por bendecirme con fortaleza, sabiduría y el valor para superar los desafíos y por iluminar mi vida con su presencia.

A mis queridos padres, cuyo amor incondicional, apoyo y sacrificio han sido fundamentales en mi vida. Gracias por creer en mí, por sus consejos y por estar siempre a mi lado. Este logro es también de ustedes.

A mi familia en general, por ser mi refugio en los momentos de dificultad y por celebrar conmigo cada pequeño triunfo. Su cariño y compañía han sido un pilar fundamental en este viaje.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han sido parte fundamental en la realización de este trabajo de grado:

A mi profesora y directora de trabajo de grado, Laura Viviana Galvis, por su orientación, paciencia y dedicación. Su conocimiento y experiencia han sido fundamentales en la culminación de este proyecto.

A mis compañeros de estudio, con quienes compartí innumerables horas de esfuerzo y aprendizaje. Juntos enfrentamos desafíos y celebramos logros. Agradezco profundamente su amistad, colaboración y las experiencias compartidas que hicieron este proceso más llevadero y enriquecedor.

A mi compañero de vida, por ser mí apoyo emocional durante todo este viaje. Gracias por tu amor y comprensión, por estar a mi lado en los momentos difíciles y por celebrar conmigo cada paso adelante.

A mis amigos, quienes me brindaron su aliento, comprensión y alegría a lo largo de este camino. Su apoyo incondicional y las risas compartidas me dieron la energía necesaria para seguir adelante.

A todos ustedes, gracias de corazón por acompañarme en este importante capítulo de mi vida.

Tabla de contenido

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Siglas y Abreviaturas

LSTM: *Long Short-Term Memory. (Memoria de corto-largo plazo)*

ANN: Artificial Neural Network **(***Red Neuronal Artificial)*

API: *Application Programming Interface (Interfaz de Programación de Aplicaciones).*

ML: *Machine Learning (Aprendizaje de máquina).*

EDA: *Exploratory Data Analysis (Análisis Exploratorio de Datos).*

IQR: *Interquartile Range (Rango Intercuartílico).*

KWh: *Kilowatt-hour (Kilovatio-hora).*

MSE: *Mean Squared Error (Error Cuadrático Medio).*

MAPE: *Mean Absolute Percentage Error (Error Porcentual Absoluto Medio).*

ESSA: *Electrificadora de Santander S.A. E.S.P.*

Glosario

API: Las interfaces de programación de aplicaciones (API del inglés *Application Programming Interface)* son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. Por ejemplo, el sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono "habla" con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono. (¿Qué es una API?, s.f.)

Interfaz: Una interfaz es el punto de interacción y comunicación entre dos sistemas, ya sea entre un usuario y un dispositivo o entre dos aplicaciones de software. En el contexto del desarrollo de software, la interfaz de usuario (UI del inglés User Interface) se refiere al espacio donde las interacciones humano-computadora ocurren. Se diseña para facilitar el uso del sistema, permitiendo que los usuarios interactúen con el de manera eficiente y efectiva. Una interfaz bien diseñada no solo mejora la experiencia del usuario, sino que también aumenta la usabilidad y accesibilidad del sistema. (Nielsen, 1993, p. X).

Keras: Keras es una API de redes neuronales escrita en lenguaje Python. Se trata de una libreriía de código abierto que se ejecuta sobre frameworks como Theano y TensorFlow. Diseñada para ser modular, rápida y fácil de usar, es una de las más utilizadas para el desarrollo y las pruebas de redes neuronales. Facilita enormemente la creación de capas para las redes neuronales o la configuración de arquitecturas complejas. (Keras, s.f.).

Matplotlib: Matplotlib es una librería de trazado utilizada para gráficos 2D en lenguaje de programación Python, es flexible y tiene gran cantidad de valores predeterminados incorporados que facilitan el trabajo de análisis de datos. Produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos. Para utilizar esta librería se deben hacer las importaciones necesarias, preparar los datos y con esto se puede comenzar a trazar una función con la ayuda de la instrucción plot().(¿Qué es matplotlib y cómo funciona?, s.f.).

Mockups: Los mockups son representaciones visuales de un producto o interfaz que muestran la estructura básica y el diseño visual de una aplicación o página web. A diferencia de los wireframes, los mockups suelen incluir elementos de diseño detallados como colores, tipografías y imágenes, aunque no son completamente funcionales. Su propósito es proporcionar una visión clara de cómo se verá el producto final antes de que se inicie el desarrollo, lo que permite identificar y solucionar problemas de diseño en las primeras etapas del proyecto (Rudd, Stern, & Isensee, 1996, p. 76).

NumPy: El paquete es conocido por su propia estructura de datos, llamado arreglo de NumPy. NumPy también permite a los desarrolladores de Python realizar en forma rápida una amplia variedad de cálculos numéricos (La guía definitiva del paquete NumPy, s.f.).

Python: Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el aprendizaje de máquina (ML del inglés machine learning). Los desarrolladores utilizan Python porque es eficiente, además de que se puede ejecutar en cualquier plataforma. Las características que presenta en su sintaxis y su tipado dinámico lo

convierten en un lenguaje ideal para scripting y para el desarrollo de aplicaciones en diferentes áreas (inteligencia artificial, ciencia de datos, desarrollo web, entre otros) (Manobanda Vega, 2020)

Pronóstico de Energía a Corto Plazo: Se refiere a la predicción de la demanda de energía eléctrica en un horizonte temporal cercano, típicamente de horas a días. Es crucial para la planificación operativa del sistema eléctrico, permitiendo ajustar la producción de energía a las necesidades reales. (Weron, 2014, p. 1030).

Preprocesamiento de Datos: Es el conjunto de técnicas aplicadas a los datos brutos para prepararlos para el análisis o modelado. Incluye la limpieza de datos, la normalización, la eliminación de valores atípicos, y la transformación de datos cualitativos en cuantitativos (Han, Pei, & Kamber, 2011, p. X).

Prototipo de Software: Un prototipo de software es una versión preliminar de una aplicación que permite explorar ideas de diseño y probar funcionalidades. En el contexto del proyecto, el prototipo ayuda a visualizar el comportamiento del modelo LSTM y a realizar ajustes antes de desarrollar la versión final. (Budde et al., 1992, p. X).

Red Neuronal LSTM (Long Short-Term Memory): Las LSTM son un tipo avanzado de red neuronal recurrente que puede aprender a largo plazo y recordar secuencias de datos. Se usan comúnmente en tareas de predicción de series temporales, como el pronóstico de la demanda

eléctrica, debido a su capacidad para manejar dependencias temporales complejas (Hochreiter & Schmidhuber, 1997, p. 1735).

Series temporales: Conjunto de datos observados en momentos sucesivos a lo largo del tiempo. Las series temporales se utilizan en el pronóstico de demanda eléctrica para capturar patrones temporales y hacer predicciones futuras basadas en datos históricos (Hyndman & Athanasopoulos, 2018, p. X).

Tensorflow: TensorFlow es una plataforma de código abierto para el aprendizaje automático y el análisis de datos. Es uno de los frameworks más utilizados en la industria y en la investigación, gracias a su capacidad para crear y entrenar modelos de aprendizaje profundo (del inglés deep learning), redes neuronales y cómputo con GPUs de manera eficiente, también permite a los usuarios crear y entrenar modelos de aprendizaje automático en una variedad de tareas, desde la clasificación de imágenes hasta la generación de texto. (Descubre TensorFlow, 2023).

Tkinter: Tkinter es una biblioteca estándar de Python utilizada para crear interfaces gráficas de usuario (GUI). Es conocida por su facilidad de uso y simplicidad, lo que la convierte en una opción popular para el desarrollo rápido de prototipos de software (Grayson, 2000, p. X).

Resumen

Título: Prototipo de software para el pronóstico de demanda de energía en la Electrificadora de Santander.*

Autor: María Fernanda Rodriguez Pulido**

Palabras Clave: Software, Pronóstico, Energía, Python, Tkinter, Redes Neuronales, Aprendizaje profundo, LSTM.

Descripción:

La proyección precisa de la demanda de energía eléctrica es crucial para garantizar la continuidad del suministro eléctrico, lo que no solo genera confiabilidad en los consumidores, sino que también es una obligación regulatoria en el sector eléctrico. Minimizar el error en la predicción de la demanda de energía es esencial para evitar posibles sanciones por parte de los entes reguladores.

Este proyecto tiene como objetivo desarrollar un modelo de pronóstico de energía a corto plazo utilizando redes neuronales con capacidades de aprendizaje a corto y largo plazo, como las Redes Neuronales de Memoria a Largo Corto Plazo (LSTM). Para ello, se emplearon datos históricos de consumo de energía proporcionados por la Electrificadora de Santander.

En este contexto, el proyecto evalúa la efectividad del modelo de Redes Neuronales LSTM. Asimismo, se desarrolló un prototipo de software utilizando Python y la librería Tkinter, que permite a los usuarios visualizar gráficamente las predicciones generadas por el modelo LST

* Trabajo de Grado
** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Ingeniería de Sistemas. Director: Laura Viviana Galvis Carreño. Ph.D. en Ingeniería Eléctrica y Computación.

Abstract

Title: Software Prototype for Energy Demand Forecasting at Electrificadora de Santander.*

Author(s): María Fernanda Rodriguez Pulido**

Key Words: Software, Forecasting, Energy, Python, Tkinter, Neural Networks, Deep Learning, LSTM.

Description:

The accurate estimation of electricity demand is crucial to ensure the continuity of power supply, which not only builds consumer trust but is also a regulatory obligation in the energy sector. Minimizing error in demand forecasting is essential to avoid potential penalties from regulatory authorities.

This project aims to develop a short-term energy forecasting model using neural networks with both short and long-term learning capabilities, such as Long Short-Term Memory (LSTM) networks. To achieve this, historical energy consumption data provided by Electrificadora de Santander was used.

In this context, the project evaluates the effectiveness of the LSTM model. Additionally, a software prototype was developed using Python and the Tkinter library, allowing users to visually display the predictions generated by the LSTM model

* Degree Project
** Faculty of Physical-Mechanical Engineering. School of Systems and Computer Engineering. Systems Engineering. Director: Laura Viviana Galvis Carreño, Ph.D. in Electrical and Computer Engineering.

Introducción

La predicción de la demanda de energía eléctrica es un proceso crucial para el funcionamiento eficiente y confiable de los sistemas eléctricos. Esta tarea es vital para garantizar que los diferentes activos del sistema eléctrico operen de manera óptima, permitiendo una adecuada planificación y gestión de los recursos energéticos. Históricamente, las predicciones de demanda se han abordado mediante una variedad de métodos, incluyendo series temporales, regresiones simples y múltiples, y modelos econométricos. Sin embargo, en las últimas décadas, el avance de la tecnología ha introducido técnicas más sofisticadas, como la inteligencia artificial, que han mejorado significativamente la precisión de estas predicciones. Entre estas técnicas destacan el aprendizaje automático, las redes neuronales artificiales, los algoritmos genéticos y la lógica difusa, cada uno con características matemáticas y computacionales específicas que ofrecen ventajas y desventajas según la aplicación y los datos disponibles (Weron, 2014, p. 1030).

Los errores en la predicción de la demanda no solo afectan la planificación operativa y los costos, sino que también impactan en la seguridad del sistema, el flujo de carga y los planes de mantenimiento preventivo. Por lo tanto, mejorar la precisión del pronóstico es fundamental para garantizar un funcionamiento fiable y económico del sistema eléctrico. Paladino (2022) comenta que con el avance de las tecnologías de procesamiento computacional, se han podido implementar algoritmos más complejos, como las redes neuronales artificiales; estas permiten aprender los patrones y

tendencias de los datos históricos para realizar predicciones más precisas, minimizando el riesgo de sobreestimación o subestimación de la demanda y las consecuencias asociadas.

El aumento continuo en el consumo de energía eléctrica destaca la importancia de proyectar adecuadamente la demanda futura, ya que esto es crucial para mantener la confiabilidad, continuidad, seguridad, y la calidad del suministro eléctrico. El uso de modelos avanzados, como las redes neuronales de Memoria a Largo Corto Plazo (LSTM), que son una variante de las redes neuronales recurrentes (RNN), ha demostrado ser especialmente efectivo para capturar patrones en series temporales, lo que es crucial en la predicción de la demanda eléctrica (Hochreiter & Schmidhuber, 1997, p. 1735). Estas redes, gracias a su capacidad para gestionar dependencias a largo plazo, ofrecen una solución robusta a los problemas asociados con la predicción de series temporales complejas.

Este trabajo se estructura en cinco capítulos. El primer capítulo, titulado "Contextualización y Fundamentación del Proyecto", se enfoca en la importancia del pronóstico de la demanda eléctrica y su relación con la eficiencia del sistema eléctrico. Se describe el problema específico que motiva la realización del proyecto, su relevancia y los objetivos generales y específicos establecidos para abordar dicho problema. Además, se describe la metodología seleccionada para el desarrollo del proyecto, proporcionando una visión clara de cómo se llevó a cabo dicho proceso.

El segundo capítulo, "Análisis y Procesamiento de Datos", se centra en la descripción de los datos de consumo histórico y otras variables relevantes utilizadas en el proyecto. Incluye una investigación de las APIs meteorológicas empleadas para obtener datos climáticos y su integración en el modelo.

En el tercer capítulo, "Diseño y Entrenamiento del Modelo", se describe la arquitectura de la red neuronal LSTM utilizada, incluyendo las capas y funciones activas que la componen. Se detalla el proceso de entrenamiento del modelo, los parámetros utilizados y el proceso de validación, así como los resultados obtenidos y la evaluación del rendimiento del modelo.

El cuarto capítulo, "Desarrollo del Prototipo de Software", aborda el diseño y las funcionalidades del software desarrollado para visualizar las predicciones. Se explica la implementación de las funcionalidades utilizando Python y Tkinter, se detallan las diferentes pantallas del software y cómo se visualizan las predicciones, y se presentan las pruebas realizadas para asegurar el correcto funcionamiento del software y los resultados obtenidos.

Finalmente, el quinto capítulo, "Conclusiones y Reflexiones", resume los hallazgos principales del proyecto y cómo estos responden a los objetivos establecidos. Se reflexiona sobre la importancia de los resultados obtenidos para la ESSA y el sector eléctrico en general, se detallan los aportes personales y profesionales del proyecto, y se ofrecen recomendaciones para futuras investigaciones o mejoras en el área de pronóstico de demanda eléctrica.

1 Generalidades

1.1 Modalidad de trabajo de grado

El presente trabajo de grado se desarrolló bajo la modalidad de práctica empresarial en la ESSA (Electrificadora de Santander S.A. E.S.P.). Durante esta práctica, se brindó apoyo en la transformación de diversos procesos clave mediante la implementación de tecnologías 4.0. Esta modalidad fue avalada por la Escuela de Ingeniería de Sistemas como opción para la culminación del proyecto de grado y la obtención del título de Ingeniero de Sistemas.

1.2 Empresa ESSA y el pronóstico de la Energía

La ESSA (*Electrificadora de Santander S.A. E.S.P.)* es una empresa clave en la gestión y distribución de energía eléctrica en el departamento de Santander, Colombia. Su misión es garantizar un suministro eléctrico eficiente y continuo a sus clientes, optimizando el uso de recursos y mejorando la calidad del servicio. En este contexto, el pronóstico de la demanda eléctrica se convierte en una función crucial para el éxito de sus operaciones y la planificación estratégica.

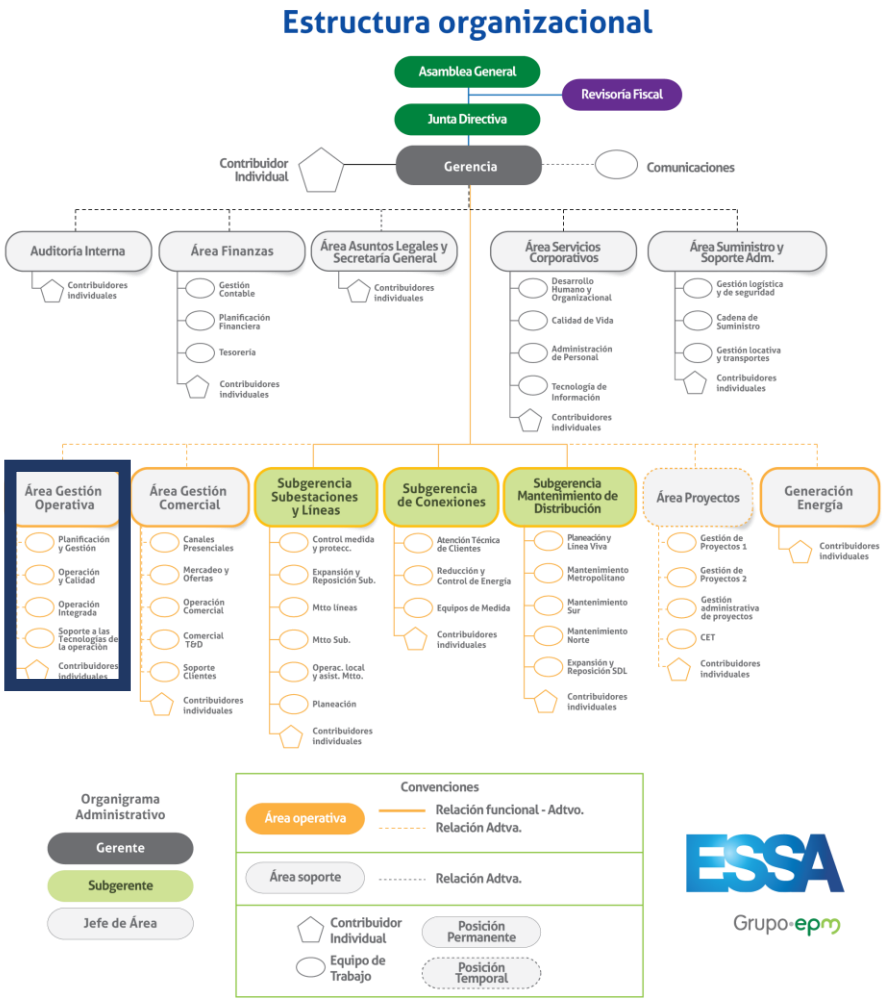
Dentro de la ESSA, el área de Operación y Estudios Eléctricos desempeña un papel vital en la elaboración de pronósticos de demanda. Este departamento es responsable de estimar la cantidad de electricidad requerida en diferentes periodos, basándose en datos históricos de consumo, condiciones meteorológicas, y otros factores que pueden influir en la variabilidad de la demanda. Los pronósticos precisos son esenciales para la correcta

planificación de la infraestructura eléctrica, la gestión eficiente de la energía y la reducción de costos operativos.

El proceso de pronóstico de la demanda de energía se desarrolla en el área operativa de la Electrificadora de Santander S.A. E.S.P. (ESSA), donde se llevó a cabo la práctica, tal como se muestra resaltado en la Figura 1, que ilustra su estructura organizacional. Entre las funciones del área operativa, se resalta el proceso que involucra la recopilación y análisis de grandes volúmenes de datos sobre el consumo eléctrico, los cuales se emplean para construir modelos predictivos que permiten anticipar las fluctuaciones en la demanda. La precisión en estos pronósticos no solo garantiza un suministro eléctrico más estable, sino que también permite a la ESSA adaptarse proactivamente a los cambios en el consumo, optimizar la asignación de recursos y reducir el riesgo de interrupciones en el servicio.

Figura 1.

*Estructura Organizacional de la ESSA. Enfoque **Área de Gestión Operativa.*** Adaptada por los autores para enfocar el área Gestión operativa. Tomado de https://www.essa.com.co/site/.



1.3 Contextualización y Fundamentación del Proyecto

La predicción de la demanda eléctrica es crucial para la planificación y operación eficiente de los sistemas eléctricos. Esta función permite anticipar el consumo de energía y ajustar la oferta para mantener la estabilidad del suministro. Tradicionalmente, los métodos estadísticos simples, como las regresiones lineales y los modelos de series temporales, eran los principales instrumentos utilizados para este propósito. Aunque efectivos en algunos contextos, estos

métodos a menudo carecían de la capacidad para capturar la complejidad y la variabilidad del consumo moderno (Weron, 2014, p. 1030).

El avance en tecnologías como el aprendizaje automático y las redes neuronales ha revolucionado la predicción de la demanda eléctrica. Las redes neuronales LSTM (Long Short-Term Memory), en particular, han mostrado un gran potencial para manejar datos secuenciales y capturar patrones a largo plazo en series temporales, mejorando así la precisión de las predicciones (Hochreiter & Schmidhuber, 1997, p. 1735). Estas tecnologías permiten el análisis de grandes volúmenes de datos históricos y en tiempo real, facilitando una comprensión más profunda de las dinámicas del consumo eléctrico.

En la actualidad, Hong y Fan (2016) enfatizan que los sistemas avanzados de predicción integran datos provenientes de diversas fuentes, como condiciones meteorológicas, eventos sociales y patrones históricos de consumo. Esta integración permite una mayor precisión en las estimaciones y es especialmente importante en un contexto donde la generación de energía se está volviendo más descentralizada y dependiente de fuentes renovables intermitentes, como la solar y la eólica. Paladino (2022) expone que las predicciones precisas no solo ayudan a equilibrar la oferta y la demanda en tiempo real, sino que también optimizan el uso de recursos energéticos, reducen los costos operativos y minimizan el riesgo de apagones.

Se anticipa que el futuro de la predicción de la demanda eléctrica estará marcado por avances tecnológicos continuos. Goodfellow, Bengio y Courville (2016) exponen que el

desarrollo de algoritmos más sofisticados y el uso de inteligencia artificial mejorarán aún más la precisión de las predicciones y permitirán una adaptación más efectiva de las estrategias de gestión de redes. Además, la incorporación de tecnologías emergentes como el Internet de las Cosas (IoT del inglés Internet of Things) proporcionará datos más detallados a nivel de dispositivos individuales, lo que permitirá personalizar aún más las predicciones y ajustarlas a las necesidades específicas de cada usuario.

Zhang, Qi y Wang (2020) discuten que estos avances no solo mejorarán la operación diaria de los sistemas eléctricos, sino que también serán fundamentales para una transición hacia un sistema energético más sostenible y resiliente. Esta transición implica una integración efectiva de las energías renovables y la adaptación a los cambios en los patrones de consumo a medida que se electrifican más sectores, como el transporte.

Para la ESSA, la predicción precisa de la demanda eléctrica es esencial para la expansión de la infraestructura eléctrica y la operación eficiente del sistema. El pronóstico de demanda, que actualmente se realiza utilizando modelos de regresión lineal en hojas de cálculo, debe ser mejorado para incluir variables adicionales, como la temperatura y el día de la semana, que afectan el consumo eléctrico. López y Martínez (2019) señalan que automatizar este proceso mediante tecnologías avanzadas permitirá una mayor precisión en las predicciones y contribuirá a mantener la eficiencia en el despacho energético del país.

En el contexto actual, el pronóstico de la demanda de energía eléctrica requiere no solo tiempo y criterio experto, sino también una cuidadosa consideración de diversos factores que pueden influir en el consumo. Las desviaciones entre el pronóstico y la demanda real pueden generar incumplimientos regulatorios, que en empresas como la Electrificadora de Santander (ESSA), resultan en pérdidas económicas y sanciones por parte de los entes de control. A pesar de los métodos tradicionales basados en regresiones lineales simples, que se llevan a cabo en hojas de cálculo y que utilizan variables independientes como la temperatura obtenida de sitios web de predicción meteorológica, las desviaciones siguen ocurriendo, evidenciando la necesidad de adoptar enfoques más avanzados.

El objetivo de este proyecto de práctica empresarial es predecir la demanda de energía eléctrica tanto diaria como semanal para la ESSA, utilizando redes neuronales profundas. Para lograr esto, se consideran variables relevantes como los días de la semana, festividades y datos históricos de demanda. Como resultado, se ha desarrollado un software en Python, este refutado en las investigaciones de Alharthi y Jebelli (2021), empleando Tkinter para la interfaz gráfica, que integra librerías avanzadas como Keras y TensorFlow para la implementación y entrenamiento de modelos de aprendizaje profundo.

2 Objetivos

2.1 Objetivo General

Desarrollar prototipo de software para pronosticar la demanda del consumo eléctrico mediante el uso de una red neuronal, con el fin de obtener mayor precisión en el pronóstico para la Electrificadora de Santander.

2.2 Objetivos Específicos

1. Realizar preprocesamiento y análisis de los datos suministrados por el equipo de operación integrada mediante el uso de librerías Python.
2. Diseñar, entrenar y evaluar la red neuronal LSTM para la predicción de la demanda de consumo eléctrico utilizando Python
3. Diseñar interfaz para la visualización y generación de la información pronosticada mediante reportes.

3 Marco de referencia

3.1 Pronóstico de la Demanda

Se define a la predicción como el proceso que intenta determinar los valores de una o varias variables a partir de un conjunto de datos. Es decir, la predicción de la demanda eléctrica es aquel proceso con el cual se determina el valor de la demanda eléctrica futura a partir de una base de datos que contempla los valores históricos de la demanda eléctrica y de las diferentes variables que influyen en el comportamiento del mismo (Reinoso, 2022, p. X).

3.1.1 Curva de la Demanda

La variación de la demanda en el tiempo también puede ser representada gráficamente mediante la curva de demanda en la cual el eje de las abscisas representa el tiempo y el eje de las ordenadas representa la demanda. Dependiendo de la duración del intervalo de tiempo considerado, se pueden observar diferentes comportamientos de la demanda. Estos gráficos permiten identificar patrones diarios, semanales, mensuales o anuales, como los picos de consumo en determinadas horas del día, las diferencias entre días laborales y fines de semana, o las variaciones estacionales a lo largo del año (Reinoso, 2022, p. X).

3.1.2 Demanda Eléctrica

Se define a la demanda como el valor promedio de la potencia eléctrica tomada por un usuario en los terminales de recepción durante un periodo de tiempo denominado intervalo de demanda, la duración de este intervalo depende de la demanda a analizar y el propósito de su estudio, puede estar especificada en segundos, minutos, horas, días, meses o incluso años. El valor promedio de la potencia durante el intervalo de demanda se obtiene al dividir la energía (kWh) acumulada durante el intervalo de demanda por la duración del intervalo en horas.

3.1.3 Factores que influyen en la demanda

La demanda de energía eléctrica de un sistema de potencia es influenciada por varios factores tales como: variables meteorológicas, socioeconómicas y demográficas. Por esto, el número de las variables que se requieren depende de la naturaleza del pronóstico, por lo tanto, estas variables deben ser seleccionadas cuidadosamente (Pérez, 2021, p. X).

3.2 Redes Neuronales

Una red neuronal es una herramienta de inteligencia artificial que se diseña para enseñar a las computadoras a procesar datos, inspirada en la forma en que lo hace el cerebro humano. Se trata de un tipo de proceso de aprendizaje profundo (del inglés machine learning), que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano, creando un sistema adaptable que las computadoras

utilizan para aprender de sus errores y mejorar continuamente. De esta forma, las redes neuronales artificiales intentan resolver problemas complejos, como la realización de resúmenes de documentos o el reconocimiento de rostros con mayor precisión (Acuerdo 1303, 2020)

El conocimiento se adquiere por la red mediante un proceso de aprendizaje y las fuerzas de conexión interneuronal, conocidas como ponderaciones sinápticas, se utilizan para almacenar el conocimiento (Hyndman & Athanasopoulos, 2018, p. X).

3.2.1 Arquitectura de las Redes Neuronales

Una red neuronal artificial (ARN del inglés Artificial Neural Network) básica tiene neuronas interconectadas en 3 capas:

Capa de entrada: La información del mundo exterior entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa.

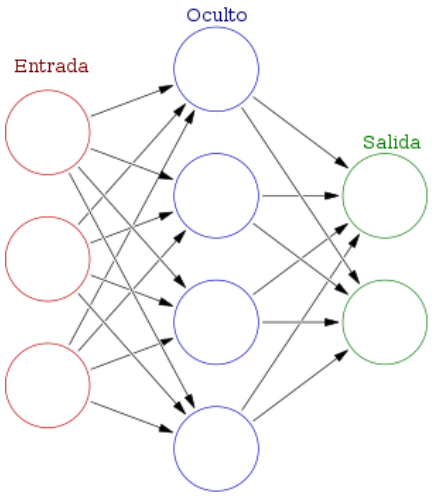
Capa oculta: Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa.

Capa de salida: La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos, ver Figura 2. Por ejemplo, si tenemos un problema de clasificación binaria (sí/no),

la capa de salida tendrá un nodo de salida que dará como resultado 1 o 0. Sin embargo, si tenemos un problema de clasificación multiclase, la capa de salida puede estar formada por más de un nodo de salida.

Figura 2.

Arquitectura de las Redes neuronales.



Nota. El grafico representa la arquitectura de una red neuronal. Tomado de ***Wikipedia. (s.f.).*** *Red neuronal artificial. En Wikipedia, la enciclopedia libre.*

Las redes neuronales profundas, o redes de aprendizaje profundo, tienen varias capas ocultas con millones de neuronas artificiales conectadas entre sí. Un número, denominado peso, representa las conexiones entre un nodo y otro. El peso es un número positivo si un nodo estimula a otro, o negativo si un nodo suprime a otro. Los nodos con valores de peso más altos tienen mayor influencia en los demás nodos (Acuerdo 1303, 2020).

3.2.2 Mecanismos de Aprendizaje de las Redes Neuronales

La ARN utiliza un vector de datos del exterior del cual aprende, en la mayoría de los casos la forma en la que se hace uso de este vector de datos caracteriza el tipo de aprendizaje (Manobanda Vega, 2020).

Existen 2 tipos de aprendizajes:

Aprendizaje supervisado: consiste en mostrar a la salida de la ARN un valor deseado con el cual comparar, con la finalidad de calcular un error y modificar los bias y pesos sinápticos.

Aprendizaje no supervisado: este tipo de aprendizaje no tiene un valor de comparación a la salida, este tipo de ARN tiene la capacidad de ajustar sus pesos sin necesidad de un agente externo que controle el desarrollo de esta.

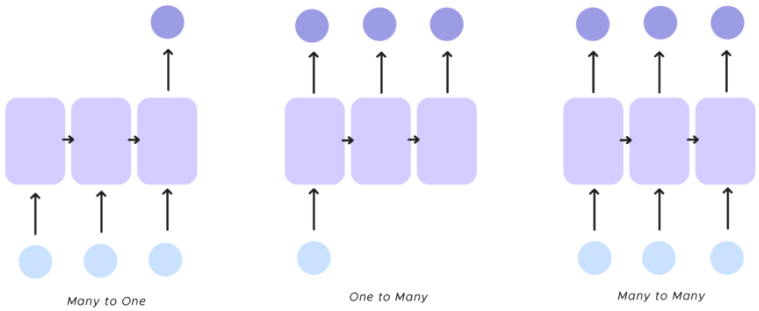
3.2.3 Redes Neuronales Recurrentes

Una red neuronal recurrente (RNN del inglés Recurrent Neural Network) es un tipo de red neuronal artificial que utiliza datos secuenciales o datos de series de tiempo. Estos algoritmos de aprendizaje profundo se utilizan comúnmente para problemas ordinales o temporales, como la traducción de idiomas, el reconocimiento de voz y subtítulos de imágenes.

Al igual que las redes neuronales feedforward y convolucionales (CNN del inglés Convolutional Neural Networks), las redes neuronales recurrentes utilizan datos de entrenamiento para aprender. Se distinguen por su "memoria", ya que toman información de entradas anteriores para utilizarse en los datos de entrada y en los resultados, como se observa en la Figura 3. (Grayson, 2000, p. X).

Figura 3

Estructura de una RNN.



Nota. Este define la estructura de una RNN. Tomado de Python and Tkinter programming. Manning Publications.

3.2.4 Criterios de Evaluación del Modelo de una Red Neuronal

Entre las métricas para llevar a cabo la evaluación de modelos de redes neuronales, se destacan las siguientes:

a) Error Cuadrático Medio (MSE: Mean Squared Error): Es la medida de error más común y viene expresada por la siguiente ecuación. El error resultante entre el valor real y el valor esperado se eleva al cuadrado para resaltar el error cometido y evitar valores negativos. Se denomina como M al número total de valores que serán evaluados y el resultado se obtiene como la suma de los errores divididos para M[1]:

$$MSE = \frac{\sum_{t=1}^{M}(y_i - \hat{y}_i)^2}{M}$$

donde:

y_i : valor esperado (real).

$\hat{y}_i$: valor estimado (proyectado).

b) Raíz del error Cuadrático Medio (RMSE: Root Mean Squared Error): El proceso es similar a la métrica anterior de MSE, con la diferencia de añadir una raíz cuadrada al resultado. La raíz se introduce con el objetivo de obtener la escala de los errores en la misma escala que los objetivos (Manobanda Vega, 2020).

La ecuación de esta métrica es la siguiente:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M}(y|i - \hat{y}_i)^2}{M}}$$

donde: - y_i: valor esperado. -$\hat{y}_i$: valor estimado. - M : número total de valores.

c) Error Absoluto Medio (MAE: Mean Absolute Error): Esta métrica es muy útil cuando se desea medir el error cometido en las mismas unidades que la serie original. La medida del error entre los valores esperados y los valores obtenidos se plantea como valor absoluto para evitar valores negativos, el resultado se obtiene como la suma de los errores cometidos dividida para el total de valores (Manobanda Vega, 2020):

Esta métrica se define de la siguiente manera:

$$MAE = \frac{\sum_{i=1}^{M} |y_i - \hat{y}_i|}{M}$$

donde: - y_i: valor esperado. -$\hat{y}_i$: valor estimado. - M : número total de valores.

d) Porcentaje de Error Medio Absoluto (MAPE: Mean Absolute Percentage Error): Esta métrica permite expresar el resultado en forma porcentual, lo que resulta de mayor utilidad que utilizar términos en valor absoluto. Se calcula el error relativo porcentual para cada proyección y luego se obtiene el promedio de estos errores relativos, esta métrica viene expresada por la siguiente ecuación (Manobanda Vega, 2020):

$$MAPE = \frac{\sum_{i=1}^{M} 100 \times \frac{|y_i - \hat{y}_i|}{y_i}}{M}$$

donde: - y_i: valor esperado. -$\hat{y}_i$: valor estimado. - M : número total de valores (Manobanda Vega, 2020)

3.3 Algoritmo Backpropagation

Es un tipo de entrenamiento supervisado que hace uso del método del gradiente descendente para realizar la propagación del error entre las salidas deseadas y las proyectadas a partir de un conjunto de datos de entrada, actualizando los pesos de las neuronas en las diferentes capas, este proceso se realiza en dos fases (Manobanda Vega, 2020):

Se ingresa el vector de entrada a la red y se propaga por las capas de la red emitiendo una señal de salida. Esta salida se compara con la salida deseada y se calcula el error cometido por la neurona de salida, este error se propaga hacia atrás hasta las neuronas de la capa oculta.

A cada neurona se le atribuye un error proporcional a su contribución en el error total de la red, y basados en este error, se ajustan los pesos sinápticos de cada neurona. El método de entrenamiento backpropagation es usado para obtener una combinación correcta de pesos de una RNA de tal manera que se minimice el error. La base de esta minimización está en la retropropagación del error entre la capa de salida y la capa oculta de tal manera

que se encuentren las derivadas parciales necesarias para minimizar este error (Manobanda Vega, 2020).

3.4 Redes neuronales LSTM

Las LSTM consisten en una serie de celdas conectadas, donde cada celda tiene la capacidad de agregar o eliminar información mediante tres puertas: la puerta de entrada, la puerta de olvido y la puerta de salida. Estas puertas controlan el flujo de información, permitiendo que las LSTM conserven información relevante y descarten datos irrelevantes de manera eficiente (Olah, 2015). Este mecanismo es lo que distingue a las LSTM de otras arquitecturas y las hace ideales para aplicaciones como el procesamiento de lenguaje natural, la predicción de series temporales y la traducción automática (Graves, 2012).

En la predicción de series temporales, por ejemplo, las LSTM son capaces de analizar patrones históricos y aprender dependencias complejas en los datos, lo que mejora la precisión de las predicciones en comparación con otros modelos tradicionales de aprendizaje automático. Esto ha llevado a su adopción en una amplia variedad de campos, incluyendo el pronóstico energético, el análisis financiero y la planificación de inventarios (Hochreiter & Schmidhuber, 1997; Graves, 2012).

4 Metodología

Para el desarrollo de este proyecto, se adoptó la metodología en cascada debido a su estructura secuencial y su capacidad para proporcionar claridad y control en el proceso de desarrollo de software, según Sommerville (2016). Se basa en una serie de fases que deben completarse de manera consecutiva antes de pasar a la siguiente, como se evidencia en la comparativa realizada por Royce (1970). En el desarrollo de software, basados en la definición de Pressman (2014), se opta por implementar la metodología en cascada establece un marco claro para gestionar y ejecutar proyectos de forma ordenada y predecible.

En este caso, el proyecto se dividió en 2 componentes principales: el desarrollo del **modelo predictivo** y el desarrollo de la **interfaz gráfica.** A pesar de que ambos componentes fueron desarrollados utilizando la misma metodología, se abordaron de forma secuencial, comenzando con el diseño y entrenamiento del modelo predictivo y, posteriormente, la creación de la interfaz gráfica.

El **modelo predictivo**, es el componente principal del proyecto y por lo tanto requirió un proceso más exhaustivo de análisis, diseño, programación y pruebas. Este modelo involucra el desarrollo de un sistema de predicción de demanda eléctrica utilizando redes neuronales profundas. La metodología en cascada se adaptó de manera efectiva para gestionar el proceso de desarrollo, cumpliendo con los objetivos planteados en el proyecto, como se muestra en la Tabla 1. Una vez completado este proceso, se procedió al desarrollo de la **interfaz gráfica**, que fue gestionada como una fase independiente pero también siguiendo la metodología en cascada. Esta

interfaz tenía como objetivo permitir a los usuarios interactuar con el sistema de predicción de manera clara y accesible.

La metodología en cascada permitió una planificación detallada y un enfoque estructurado, garantizando el cumplimiento de cada fase del proyecto, desde el análisis inicial hasta la entrega final. Este enfoque facilitó la entrega de resultados tangibles en cada fase, permitiendo la validación continua y la integración de avances en el desarrollo del software, tal como expone Brooks (1987).

El proyecto se realizó en las siguientes fases, tal como se muestra en la Figura 4.

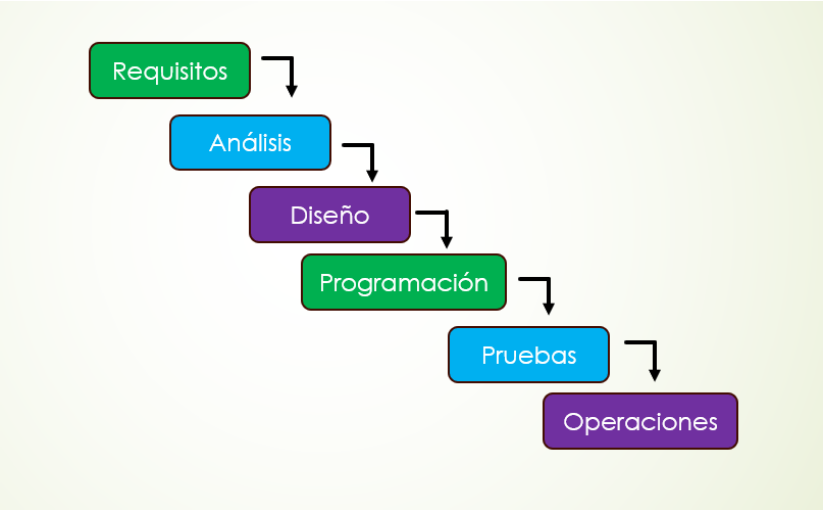
4.1 Modelo predictivo

Fase 1: Requisitos, este análisis fue esencial para establecer las bases del diseño y garantizar que todos los requisitos fueran comprendidos y documentados adecuadamente.

a. Definición de Requisitos del Sistema: En esta fase se realizó una recopilación de los requisitos tanto funcionales como no funcionales del sistema, con el fin de establecer una base sólida para el diseño y desarrollo del pronóstico de demanda.

Figura 4.

Modelo en cascada.



Nota. Esta Figura describe las fases que integran el modelo de cascada.

Fase 2: Análisis, la cual consistió en la recopilación y el manejo o pre-procesamiento de los datos, basado en Royce (1970).

a. Recopilación y Preparación de los Datos: En esta fase se recopilaron los datos históricos necesarios para el entrenamiento del modelo, y se realizó un preprocesamiento adecuado de los mismos.
    i. Recopilación de Datos: Recolección de datos históricos de demanda eléctrica proporcionados por la empresa.
    ii. Recolección de datos exógenos, como temperaturas, días festivos.

b. Preprocesamiento de Datos:
    i. Limpieza de datos: eliminación de valores nulos o erróneos.
    ii. Segmentación de los datos en conjuntos de entrenamiento y prueba

Fase 3: Diseño, En esta fase se presentó el desarrollo del modelo predictivo con LSTM, en esta fase se diseñó la red neuronal, que es el núcleo del sistema de predicción.

a. Diseño del Modelo: Definición de la arquitectura de la red LSTM, especificando el número de capas, neuronas, función de activación y método de optimización.
b. Selección del horizonte de predicción (diario, semanal).

Fase 4: Programación, que corresponde a la implementación y entrenamiento del modelo:

a. Entrenamiento del Modelo:
    i. Entrenamiento del modelo con los datos históricos preprocesados.
    ii. Evaluación del modelo mediante métricas de error.

Fase 5: Validación y Pruebas del Modelo, en esta fase se realizaron pruebas exhaustivas del modelo entrenado para garantizar su precisión y robustez.

a. Pruebas de Validación:
    i. Validación del modelo con datos no vistos (conjunto de prueba).
    ii. Comparación de los resultados del modelo con los datos reales para evaluar su precisión.
    iii. Ajustes adicionales en el modelo si los resultados no cumplen con los criterios de precisión esperados.

4.2 Interfaz gráfica (GUI)

Para esta etapa se desarrolló una interfaz gráfica que permite a los usuarios interactuar con el sistema de predicción de demanda y visualizar los resultados de manera clara y accesible, integrando los modelos de aprendizaje profundo necesarios para la predicción de la demanda eléctrica, basándose en el proceso descrito por Royce (1970).

Fase 1: Requisitos y Análisis del Sistema (Interfaz Gráfica). Una vez que el modelo predictivo estuvo validado, se recopilaron los requisitos específicos para el desarrollo de la interfaz gráfica (GUI).

a. Definición de Requisitos para la GUI:
    i. Identificación de las funcionalidades que la interfaz debía ofrecer, como la selección de fechas y la visualización de gráficos.

Fase 2: Diseño de la interfaz gráfica (GUI), En esta fase se diseñó la interfaz gráfica.

a. El diseño debe permitir al usuario seleccionar las fechas de inicio y fin para el pronóstico diario o semanal.
b. Integración de gráficos para la visualización de los resultados del pronóstico.

Fase 3: Implementación y pruebas. Una vez diseñada, la interfaz fue desarrollada y sometida a pruebas para asegurar su funcionamiento correcto y su capacidad de interactuar con el modelo predictivo.

a. Desarrollo de la interfaz implementando Tkinter.
b. Pruebas funcionales y de usabilidad para asegurar que la interfaz cumpliera con los requisitos establecidos y permitiera una visualización clara de los resultados del pronóstico.

Fase 4. Documentación y entrega de la herramienta.

El contenido de la Tabla 1 reporta el cumplimiento de los objetivos específicos del proyecto, resaltando las acciones implementadas para el cumplimiento de cada uno de ellos. Asimismo, se indica la sección del documento en la que se detalla lo desarrollado por cada objetivo.

Tabla 1.

*Cumplimiento de objetivos***.**

Objetivo	**Cumplimiento**	**Lugar en el documento**
Objetivo específico 1	Se dió cumplimiento al análisis de los datos suministrados por el equipo de operación integrada, mediante el uso de librerías Python.	Capítulo 4. Recolección y procesamiento de los datos

Objetivo específico 2	Se dió cumplimiento mediante el uso de librerías de Python, para la implementación de la red neuronal. El desarrollo de simulaciones y la evaluación constante de la red permitió proponer un modelo eficiente en la tarea de predicción de la demanda de energía.	Capítulo 4. Diseño, Arquitectura y Entrenamiento
Objetivo específico 3	Se dio cumplimiento a través de un archivo .py alojado en una carpeta designada, el cual contiene la construcción del prototipo. Los encargados del proyecto pueden acceder y ejecutarlo directamente para su compilación y uso.	Capítulo 4. Desarrollo del Prototipo de Software

Nota. Esta tabla muestra cómo se realizó el cumplimiento de los objetivos del proyecto.

5 Desarrollo de la práctica

5.1 Análisis y requisitos del software

Se inició el estudio sobre cómo se llevaba a cabo el pronóstico de la demanda de energía dentro de la organización. Se consultó con los funcionarios de la empresa sobre los procedimientos actuales, los datos que necesitaban proporcionar y cómo se utilizaban esos

datos para anticipar las fluctuaciones en el consumo eléctrico. El objetivo fue definir claramente qué debía entregar el sistema para apoyar estos pronósticos, identificando tanto los requisitos funcionales como los no funcionales descritos en la Tabla 2 y Tabla 3. Los funcionarios involucrados proporcionaron información clave para garantizar que el sistema cumpliera con las necesidades específicas del área operativa y del equipo técnico.

Tabla 2.

Requerimientos funcionales.

Lista de requerimientos funcionales
I. El modelo, debe integrar variables exógenas, como la temperatura o el día de la semana, que afecten la demanda de energía.
II. El desarrollo de este proyecto, debe entrenar un modelo de predicción utilizando redes neuronales LSTM.
III. La red neuronal debe generar pronósticos a corto plazo, diferenciando entre pronósticos diarios y semanales.
IV. La interfaz debe permitir visualización de la predicción de consumo eléctrico.
V. El margen de error de la predicción pronosticada, debe ser menor al 5%

Nota. Definición de los requerimientos funcionales del proyecto.

Tabla 3.

Requerimientos no funcionales.

Lista de requerimientos no funcionales
I. El diseño debe ser escalable para manejar grandes volúmenes de datos.
II. Debe ser accesible y usable por usuarios técnicos y no técnicos.
III. La interfaz gráfica debe ser intuitiva y fácil de usar para usuarios técnicos y no técnicos.

Nota. Definición de los requerimientos no funcionales del proyecto.

5.2 Análisis y Procesamiento de los datos

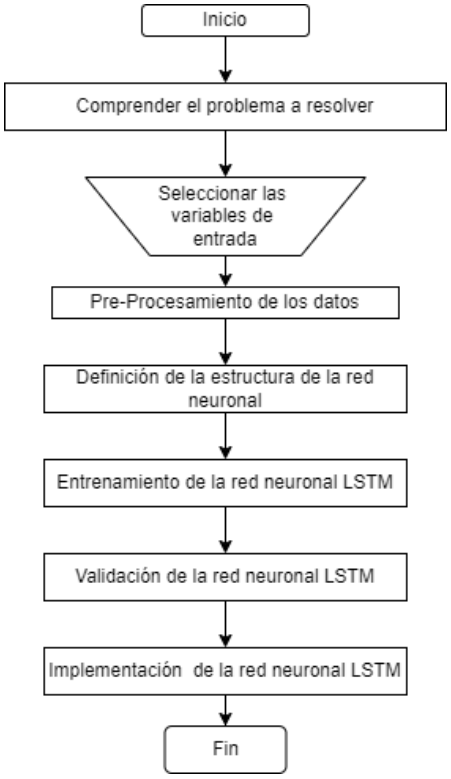
En esta sección se aborda el análisis y procesamiento de los datos proporcionados por el personal del área de Operación y Estudios Eléctricos de la Electrificadora de Santander, que corresponde a la segunda fase planteada en la metodología, el cual consiste en la recolección de los datos históricos de consumo eléctrico, así como de otras variables relevantes que puedan influir en la demanda, como las condiciones meteorológicas y los eventos sociales. Estos datos son esenciales para entender el comportamiento del consumo eléctrico a lo largo del tiempo y para identificar patrones significativos.

El análisis de los datos implica una serie de pasos, los cuales incluyen la limpieza, transformación y exploración de la información contenida en los datos, los cuales se evidencian en la Figura 5. Para este tratamiento se llevó a cabo un análisis estadístico detallado para identificar

correlaciones y tendencias en los datos, lo que permite ajustar y afinar los modelos predictivos. Fue necesario también la investigación de APIs meteorológicas, las cuales juegan un papel crucial en esta etapa, ya que proporciona datos adicionales necesarios para mejorar la precisión de las predicciones.

Figura 5.

Diagrama de Diseño y entrenamiento de la red neuronal.



Nota. Esta figura describe las actividades del proceso de desarrollo de una Red Neuronal.

5.2.1 Recolección y procesamiento de los datos

En esta etapa, se utilizó una hoja de cálculo de Microsoft Excel, denominada "Históricos.xlsx", como la fuente principal de datos (ver Figura 6). Este archivo contiene registros históricos de carga eléctrica para una región específica, cubriendo el periodo desde 2010 hasta 2023. La estructura del archivo incluye una columna para la fecha, una columna para el día, y una columna que identifica el tipo de día (por ejemplo: festivo, pandemia, especial), lo que permite definir días atípicos que pueden generar variaciones en el consumo de energía. Adicionalmente, se incluyen 24 columnas que representan el consumo horario para cada uno de los periodos del día, un total de consumo diario y una columna de temperaturas registradas. Además, el archivo detalla el consumo total dividido por las áreas en las que ESSA presta servicio, incluyendo el consumo total de 'DEMANDA METRO', 'DEMANDA NORTE', 'DEMANDA SAN GIL', y 'DEMANDA BARBOSA', así como las temperaturas correspondientes de cada área: 'TEMP METRO', 'TEMP BARRANCA', 'TEMP SAN GIL' y 'TEMP BARBOSA', como se evidencia en la Figura 7.

El conjunto de datos se dividió en dos conjuntos principales: datos de entrenamiento y datos de prueba, como se encuentra distribuido en la Tabla 4. Esta división es crucial para evaluar la capacidad del modelo para generalizar a nuevos datos. Los datos de entrenamiento se utilizan para ajustar los parámetros del modelo, mientras que los datos de prueba se reservan para validar el rendimiento del modelo y verificar su capacidad para hacer predicciones precisas sobre datos no vistos.

El conjunto de datos históricos proporcionado es lo suficientemente extenso para permitir que el modelo aprenda las dependencias y patrones de carga eléctrica a lo largo del tiempo. Esta amplitud temporal ayuda a mitigar problemas comunes en el modelado, como el sobreajuste (overfitting) y el subajuste (underfitting). El sobreajuste ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos, mientras que el subajuste se produce cuando el modelo no captura adecuadamente la estructura subyacente de los datos. La amplitud y la diversidad temporal de los datos históricos aseguran que el modelo propuesto pueda capturar las variaciones y tendencias a lo largo del tiempo, mejorando su capacidad predictiva y reduciendo el riesgo de estos problemas.

Figura 6.

Estructura de los datos Históricos por periodo de 24 horas. Fuente: Autor.

FECHA	DIA	Tipo_Dia	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
2/8/2010	lunes	Lunes	168,2	178,1	198,7	197,8	228,7	257,5	273,1	288,5	294,3	282,2	287,9	301	304,21	303,5	305,6	342,3	351,5	328,6	299,8	263,4	219
3/8/2010	martes	Martes	186,4	195,7	215	215,1	246,8	267,8	277,9	292,4	301,2	279,3	283,3	298	304,4	302,6	302,9	344,5	352,1	328	297,5	255,7	216,3
4/8/2010	miércoles	Miercoles	186,3	194,5	206,2	210,8	238,8	265	280,3	297,2	304,9	291,1	297,5	312,6	315,99	313,5	311,8	347	355,5	335,2	305,4	265,3	225,1
5/8/2010	jueves	Jueves	188,7	197	214,2	214	247,9	277,3	292,3	307,8	312,5	293,3	299,3	313,8	316,74	315,8	317,8	348,3	354,2	328,6	301,8	262,1	222,1
6/8/2010	viernes	Viernes	190,3	199,5	214,6	214,4	243,9	268,9	282,8	294,6	300,2	278,8	283,9	299,2	299,67	295,9	303,8	339,7	341,8	316,8	291,6	259,9	222
7/8/2010	sábado	Festivo	183,6	185,7	185,2	181,1	198,3	213,5	227,3	232,5	243,5	242,4	238,4	238,6	240,29	237,8	244,6	292,8	305,1	293	271,6	247,7	217
8/8/2010	domingo	Domingo	173,3	175,2	174,7	167,7	181,4	197,5	209,8	219,5	226,5	227,6	225,6	223,2	222,42	223	234,9	280,1	294,1	286,2	263,2	231,5	204,4

Figura 7.

Estructura de los datos Históricos por periodo de 24 horas, áreas de servicio. Fuente: Autor.

FECHA	DIA	Tipo_Dia	DEMANDA METRO	DEMANDA NORTE	DEMANDA SAN GIL	DEMANDA BARBOSA	TEMP METRO	TEMP BARRANCA	TEMP SAN GIL
2/8/2010	lunes	lunes	3462,538424	1992,335283	407,7558946	343,9603979	24,09	28,32	23,72
3/8/2010	martes	martes	3538,666861	2036,139381	416,7209413	351,5228172	21,88	27,85	21,62
4/8/2010	miércoles	miércoles	3593,417299	2067,642635	423,1684694	356,9615965	23,94	28,42	23,62
5/8/2010	jueves	jueves	3638,934807	2093,833287	428,528708	361,4831984	25,18	29,13	25,03
6/8/2010	viernes	viernes	3535,336311	2034,222992	416,3287287	351,1919683	23,49	27,27	23,33
7/8/2010	sábado	Festivo	3070,130532	1766,544839	361,5451059	304,9795237	23,92	28,87	23,47
8/8/2010	domingo	domingo	2899,151173	1668,163777	341,4102127	287,9948376	22,72	27,77	22,54
9/8/2010	lunes	lunes	3517,645906	2024,04398	414,2454689	349,4346451	23,66	27,73	23,01
10/8/2010	martes	martes	3636,017089	2092,15444	428,1851114	361,1933592	23,91	27,99	24,26
11/8/2010	miércoles	miércoles	3653,691497	2101,273461	430,2558887	362,9401538	24,79	28,28	24,99

5.2.1.1 API Meteorológica

Para mejorar la precisión del modelo predictivo, se realizó la búsqueda e integración de datos históricos de temperatura al conjunto de datos de demanda eléctrica. Esta adición es crucial, ya que la temperatura es un factor influyente significativo en el consumo eléctrico.

Se recopilaron datos de temperatura histórica para cada una de las 24 horas del día, etiquetados desde "T1" (correspondiente a la hora cero del día) hasta "T24" (correspondiente a las 11 de la noche). La incorporación de esta información permite al modelo captar mejor las variaciones diarias de temperatura y su impacto en el consumo eléctrico.

Además, se llevó a cabo un estudio detallado sobre una API específica para la obtención de datos históricos de temperatura. La API seleccionada debe facilitar la integración de los datos de temperatura históricos con el conjunto de datos de consumo, optimizando así la calidad y precisión de las predicciones del modelo.

5.2.1.1.1 API Open-Meteo

Open-Meteo es una API meteorológica de código abierto, que ofrece acceso gratuito a sus APIS para uso no comercial, se basan en protocolos HTTP ampliamente adoptados y usan la simplicidad de datos en formato JSON, no se requiere de clave API para consumir sus servicios. (Open-Meteo, n.d.)

Algunas características de Open-Meteo son las siguientes (Open-Meteo, n.d.):

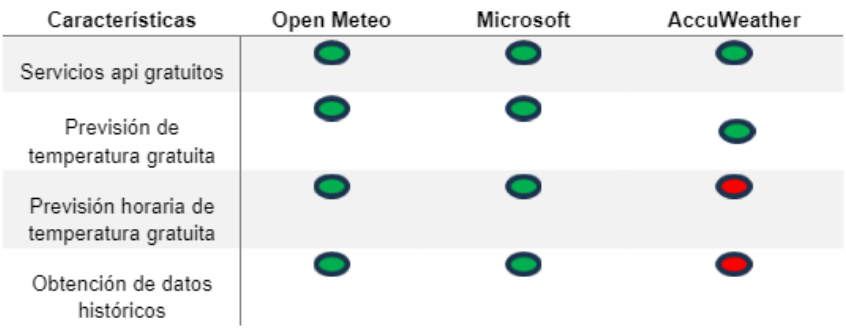
- Open-Meteo cuenta con modelos meteorológicos basados en una gran cantidad de datos en tiempo real, incluidas mediciones de diversas fuentes, como aviones, boyas, sistemas de radar y satélites. Al incorporar estos datos diversos y completos, las predicciones climáticas numéricas brindan un análisis más profundo que las estaciones meteorológicas tradicionales, lo que resulta en pronósticos más precisos.
- Reúne modelos meteorológicos locales y globales provenientes de servicios meteorológicos nacionales de renombre, lo que indica que provee pronósticos de cualquier parte del mundo. Entre los servicios nacionales están Deutscher Wetter Dienst (DWD), Administración Nacional Oceánica y Atmosférica (NOAA), Meteofrance y el Centro Meteorológico Canadiense (CMC).
- Está diseñada para brindar información meteorológica más precisa para cualquier ubicación, selecciona modelos meteorológicos de mayor resolución disponible.
- Brinda datos meteorológicos por hora para un pronóstico de 7 días. Los primeros 2-3 días del pronóstico se calculan utilizando modelos de alta resolución, ofreciendo predicciones detalladas.
- En Open-Meteo se tiene acceso a más de 80 años de datos meteorológicos por hora, que cubren cualquier ubicación de la tierra, con una resolución de 10 kilómetros.

Las características de esta API se resumen en el cuadro comparativo de la Figura 8, donde se contrasta con otras API estudiadas, como la API meteorológica de Microsoft y AccuWeather. Este

cuadro permite evaluar de manera clara las diferencias en términos de precisión, disponibilidad de datos históricos, capacidad de integración, y la cobertura geográfica que ofrece cada una. Además, se analizan aspectos clave como la frecuencia de actualización de los datos, los costos asociados y el nivel de personalización que cada API brinda para adaptarse a las necesidades específicas del pronóstico de demanda de energía.

Figura 8

Gráfico de las alternativas de la API Open-Meteo.



Nota. Este grafico muestra la comparativa entre varias APIS Meteorológicas, como lo son AccuWeather, Microsoft y Open Meteo

El código implementado en la Figura 9, se encarga de obtener los datos históricos de temperatura proporcionados por la API de Open Meteo, desglosándolos en 24 periodos correspondientes a cada hora del día. Estos datos, obtenidos en tiempo real o bajo demanda, son fundamentales para alimentar el modelo de pronóstico, ya que permiten capturar las variaciones horarias de temperatura, las cuales impactan directamente en el consumo energético. Al integrar

esta información por periodos, se garantiza que el modelo predictivo de la demanda de energía tenga en cuenta las fluctuaciones climáticas detalladas a lo largo de todo el día.

Figura 9

Obtención de temperaturas API open meteo.

```
from ast import arguments
import requests
import json
url= 'https://archive-api.open-meteo.com/v1/archive?latitude=7.1254&longi-
tude=-73.1198&start_date=2015-01-01&end_date=2023-10-04&hourly=tempera-
ture_2m,apparent_temperature&timezone=America%2FChicago'
response = requests.get(url)
print(response)
response_json = json.loads(response.text)
print(response_json)
import pandas as pd
data = {
    "time": response_json['hourly']['time'],
    "temperature": response_json['hourly']['temperature_2m'],
    "temperatura_aparente":response_json['hourly']['apparent_temperature']
}

dataframe= pd.DataFrame(data)
dataframe.head(30)
```

Nota. En este grafico se muestra como se realizó el consumo de temperaturas a la API Open Meteo.

Lo que se busca, es extraer los datos de temperatura proporcionados por la API y añadirlos a la tabla de datos históricos, de manera que se enriquezca el conjunto de información utilizado para el pronóstico de la demanda de energía. La integración de estos datos se realiza de forma estructurada y se refleja en la tabla completa, como se evidencia en la Figura 10, optimizando así la precisión y el alcance del modelo predictivo.

Figura 10.

Gráfico de construcción de la data con las temperaturas. Fuente: Autor.

	DIA	TIPO_DIA	Evento	P1	P2	P3	P4	P5	P6	P7	...	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23
2015-01-02	viernes	viernes	Ninguno	236.59	225.57	218.99	214.30	213.99	215.84	212.52	...	27.6	27.7	26.7	26.1	25.1	24.3	22.5	20.9	19.6	18.3
2015-01-03	sábado	sábado	Ninguno	256.04	241.70	232.29	225.11	223.23	226.28	217.70	...	28.3	26.5	26.3	25.7	24.5	23.4	21.6	20.1	18.7	17.6
2015-01-04	domingo	domingo	Ninguno	255.15	242.25	231.67	222.14	219.71	217.51	203.69	...	27.3	26.8	25.9	25.6	24.3	23.1	21.7	20.3	19.0	17.9
2015-01-05	lunes	lunes	Ninguno	241.93	229.83	222.53	217.70	219.67	226.99	226.41	...	28.7	27.2	25.9	25.5	24.4	23.1	21.6	19.9	18.5	17.2
2015-01-06	martes	martes	Ninguno	258.61	243.62	235.34	230.57	230.34	234.31	232.70	...	28.3	27.2	25.8	25.0	24.1	22.7	20.9	19.4	18.0	17.0

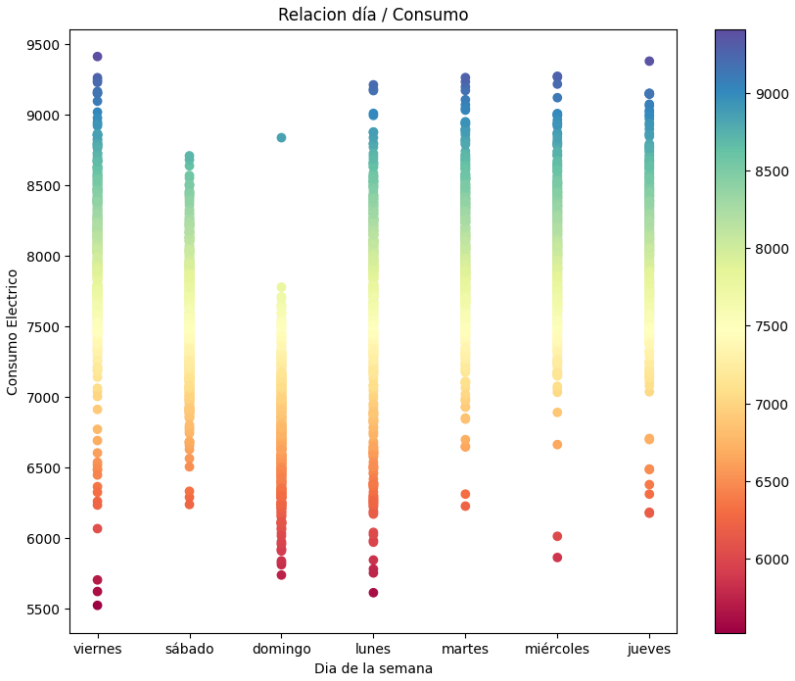
Nota. En esta figura se evidencia la estructura completa con el consumo de la temperatura por las 24 horas del día.

5.2.2 Análisis Exploratorio de Datos (EDA)

En este proyecto, se realizaron diversos análisis para evaluar el comportamiento del consumo de energía en función de diferentes variables. Se analizaron los patrones de consumo por día de la semana y su relación con la temperatura, utilizando gráficos de dispersión para visualizar las posibles correlaciones entre estas variables. Además, se llevaron a cabo análisis de series temporales que evidenciaron el comportamiento del consumo de energía a lo largo de un mes en distintos años, así como el consumo diario en periodos específicos. Para profundizar en las relaciones entre las variables, se generó un gráfico de correlación que permitió identificar la fuerza y dirección de las asociaciones entre el consumo y factores como la temperatura. También se implementó un análisis de clustering o agrupamiento mediante el algoritmo K-means, con el objetivo de agrupar los datos en función de sus similitudes, identificando patrones que pueden influir en la demanda energética. Estos análisis proporcionan una base sólida para la creación de un modelo predictivo preciso y robusto.

Figura 11.

Gráfico de dispersión de los días de la semana por el consumo eléctrico en kWh por día.

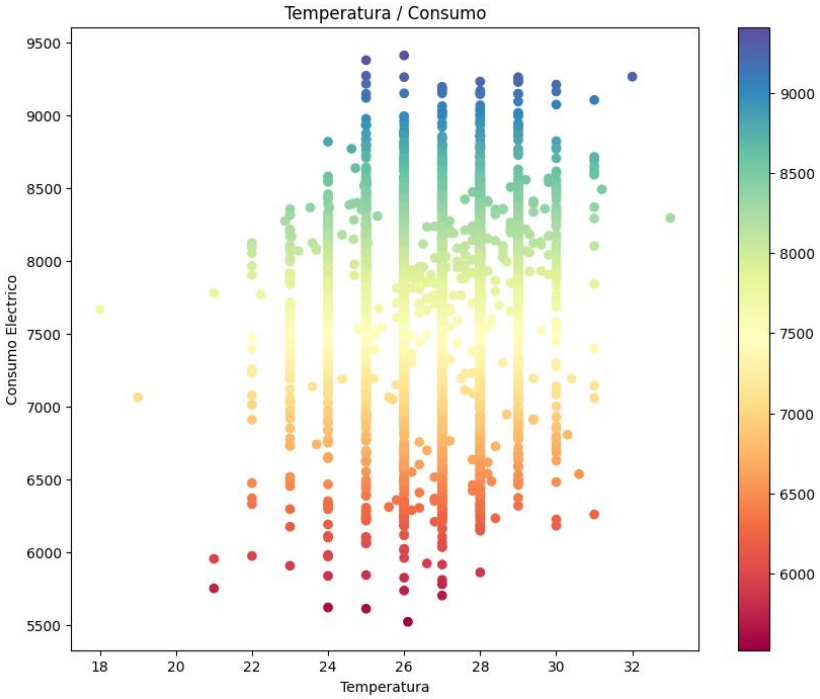


El gráfico de dispersión en la Figura 11, revela que el consumo eléctrico presenta una variabilidad significativa a lo largo de la semana de manera homogénea, aunque ciertos días (como sábado) tienden a presentar menor variabilidad en el consumo. También se observa que los días **lunes** y **viernes,** muestran una mayor amplitud en su rango de consumo**,** mientras que el día **sábado** presenta una concentración en niveles intermedios del consumo. La incorporación de coloración adicional en el gráfico añade una capa extra de información, facilitando la identificación visual de los días con los mayores y menores niveles de consumo, como lo son el día viernes, correspondiente a un mayor consumo y el día domingo, que corresponde al día de menos consumo. Esta representación visual permite una interpretación más clara de las

tendencias diarias, ayudando a destacar patrones y anomalías en el comportamiento del consumo eléctrico.

Figura 12.

Gráfico de dispersión de las temperaturas por el consumo eléctrico en kWh por día



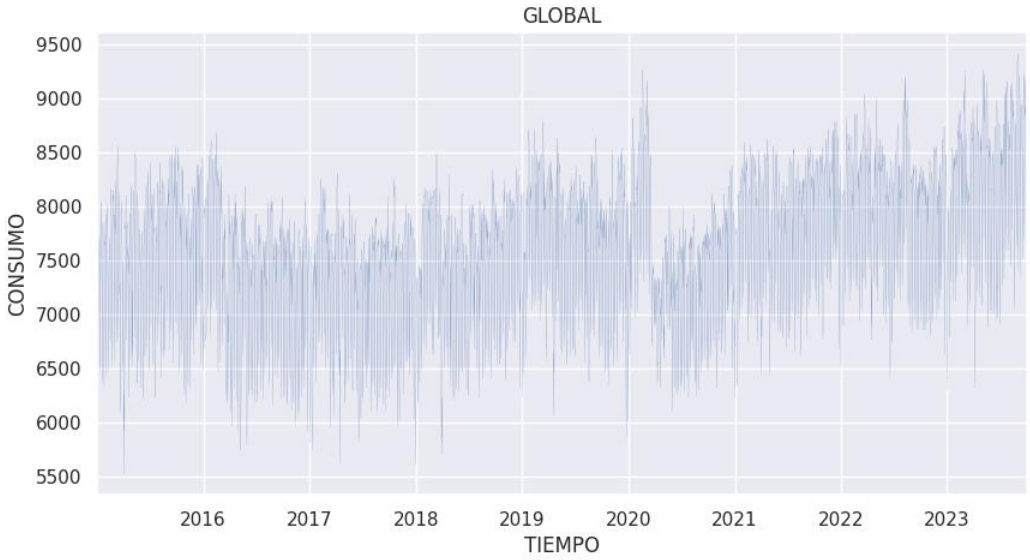
Por su parte, el gráfico en la Figura 12, reporta los valores de temperatura frente al consumo eléctrico, el cual evidencia que la mayoría de los puntos se agrupan en un rango de temperatura entre 22°C y 30°C. A medida que la temperatura aumenta desde 18°C hasta aproximadamente 24°C, se observa un incremento en la dispersión de los puntos, lo que indica una mayor variabilidad en el consumo eléctrico. Sin embargo, a partir de alrededor de 26°C, esta variabilidad parece disminuir, con el consumo manteniéndose en niveles elevados.

Aunque no existe una relación estrictamente lineal entre la temperatura y el consumo, se identifica una tendencia que sugiere un mayor consumo eléctrico en un rango de temperaturas intermedias, específicamente entre 25°C y 30°C. En este intervalo, los puntos de consumo más alto están representados con colores fríos, indicando niveles de consumo más elevados.

En contraste, a temperaturas extremas, ya sean bajas (cercanas a los 18°C) o altas (superiores a los 30°C), el consumo se mantiene en niveles moderados, con algunos picos elevados. Esto se debe a la falta de información en esos rangos de temperatura, ya que en la zona de estudio estas condiciones no son comunes, limitando la cantidad de datos disponibles. La distribución sugiere una forma de campana invertida, lo que indica que el consumo eléctrico aumenta con la temperatura hasta alcanzar un punto óptimo, para luego estabilizarse o disminuir ligeramente.

Figura 13.

Gráfico de serie temporal de los últimos años con relación al consumo de energía por día kWh.



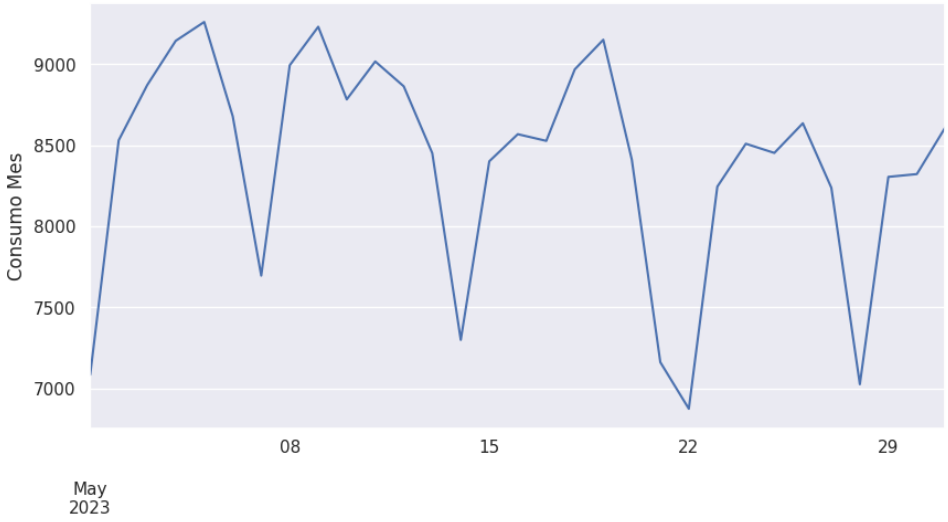
La Figura 13 ilustra la evolución del consumo de energía eléctrica a lo largo de los años, mostrando un patrón de fluctuaciones significativas desde 2016 en adelante. El gráfico presenta el consumo diario registrado cada año, destacando cómo varía la demanda de energía a lo largo del tiempo.

En la visualización se evidencia un incremento general en el consumo de energía. Las fluctuaciones diarias y anuales reflejan cambios en la demanda eléctrica que pueden estar asociados con diversos factores como las variaciones estacionales, eventos climáticos extremos y cambios en el comportamiento de los consumidores. Un ejemplo de lo mencionado anteriormente, es el pico significativo de consumo registrado en 2020 (ver Figura 13), coincidiendo con la pandemia de COVID-19. Este aumento se debió en gran parte a la mayor permanencia de las personas en sus hogares, así como al incremento en el uso de dispositivos

electrónicos para trabajo remoto, educación virtual y entretenimiento, lo que alteró de manera considerable los patrones habituales de demanda eléctrica.

Figura 14.

Gráfico de serie temporal del mes de mayo con relación al consumo de energía por día kWh.

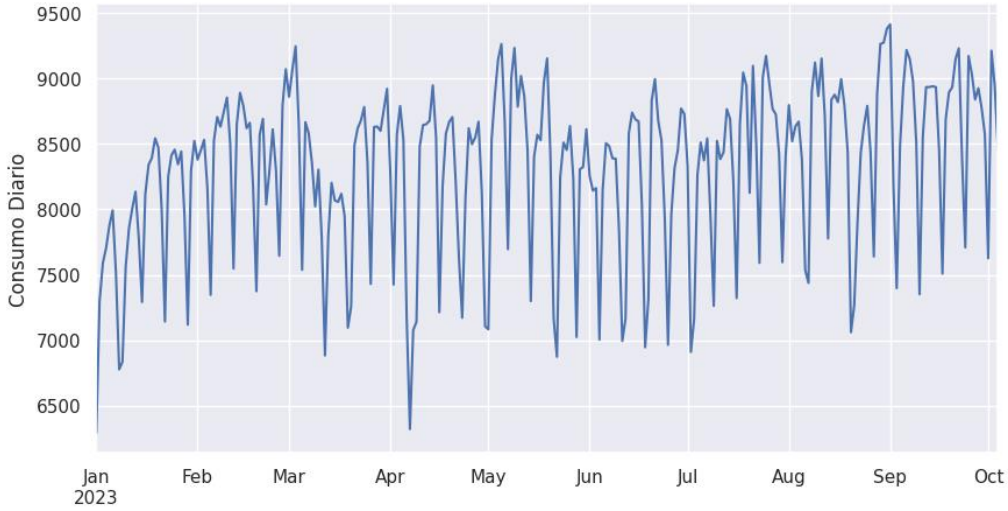


El gráfico en la Figura 14 revela patrones de fluctuación regular en el consumo eléctrico para el mes de mayo de 2023. Se observan picos de mayor consumo en los días **5, 19 y 26**, los cuales corresponden al día viernes, y según la gráfica de días (ver Figura 11), los viernes se presenta mayor consumo, lo que establece una clara relación entre el día de la semana y el aumento en la demanda eléctrica. Por otro lado, los puntos bajos correspondientes a los días **7, 14, 21 y 28**, que corresponden al día domingo, nuevamente muestran menores consumos, lo que reafirma la relación entre los días domingo y el bajo uso de energía. En este contraste, el 22 de

mayo destaca por una caída notable en el consumo, situándose por debajo de los 7000 kWh/día. Este descenso en el consumo podría ser atribuible a varios factores, como cambios en la actividad económica, condiciones climáticas inusuales o eventos específicos que afectaron la demanda de energía en esa fecha en particular.

Figura 15.

Gráfico de serie temporal de los meses del año 2023 en relación con el consumo de energía por día kWh.



El gráfico en la figura 15, revela una tendencia general de aumento en el consumo diario a lo largo del año 2023, destacando fluctuaciones regulares que sugieren posibles variaciones semanales. Se identifican picos de consumo significativos en los meses de marzo y septiembre, mientras que se observan caídas notables en abril y mayo, lo cual podría estar relacionado con eventos específicos o cambios en el comportamiento de consumo durante esos períodos,

coincidiendo con los días de mayor consumo, como los viernes, y los de menor consumo, como los domingos.

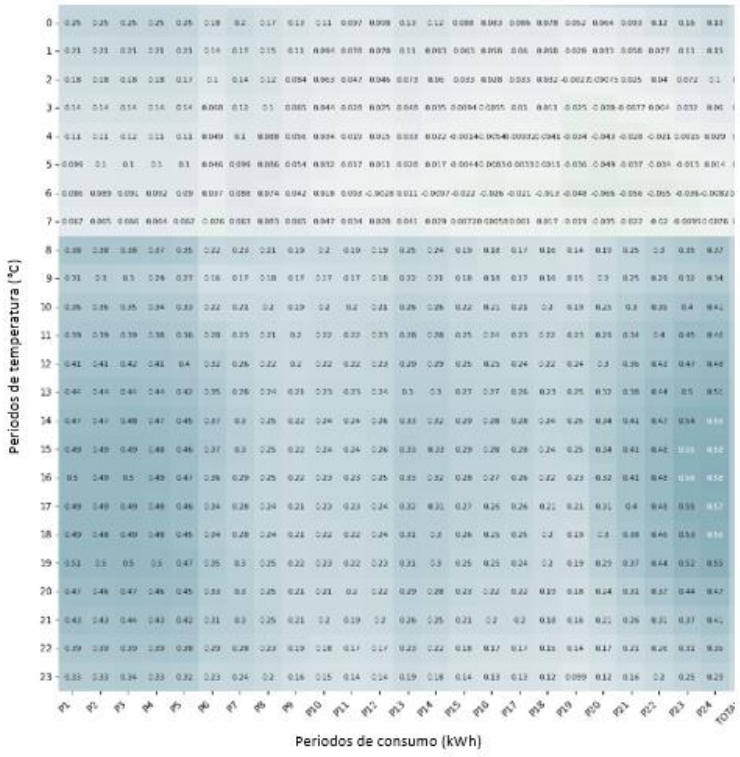
Aunque el consumo presenta variaciones a lo largo del año, no se detecta una estacionalidad clara y pronunciada. Sin embargo, se aprecia una notable volatilidad en meses como mayo y junio, donde los patrones de consumo parecen ser más erráticos. Estos cambios en el consumo podrían estar influenciados por factores como condiciones climáticas, eventos especiales, o variaciones en la demanda energética.

Este análisis temporal proporciona una visión detallada de las fluctuaciones en el consumo eléctrico, permitiendo una comprensión más profunda de los patrones de demanda y facilitando la toma de decisiones para la gestión eficiente de la energía.

En el análisis de correlación de los datos que se presenta a en la Figura 16, se observa una baja correlación entre el consumo eléctrico y la temperatura, lo que sugiere una dependencia débil. Esto indica que, aunque podría existir una relación entre ambas variables, la correlación lineal no logra capturarla de manera efectiva. La relación entre el consumo y la temperatura puede no ser lineal, lo que implica que otros tipos de relaciones podrían estar presentes, pero no se reflejan adecuadamente en la correlación lineal utilizada.

Figura 16.

Gráfico de correlación de los datos de temperatura y el consumo de energía periodo a periodo (24 horas del día).



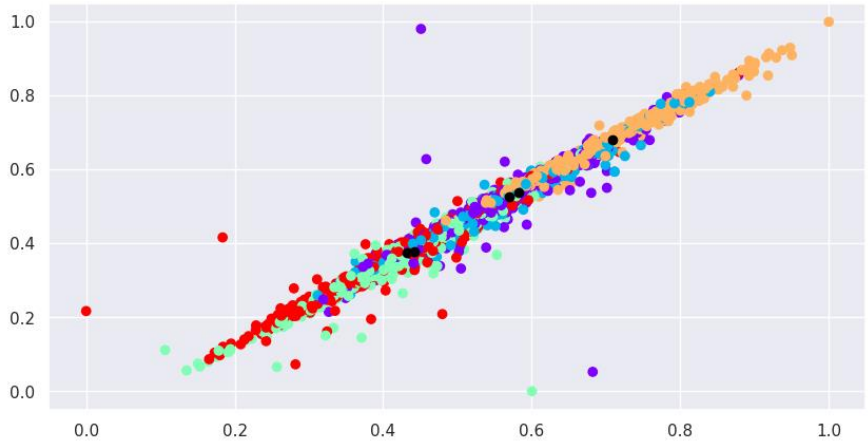
Por último, se aplicó el algoritmo K-Means para la identificación de datos atípicos en el conjunto de datos. Este método de agrupamiento (en inglés clustering) divide los datos en cinco grupos (en inglés clusters), permitiendo observar cómo se distribuyen los puntos en relación con los centros de los mismos. Esta elección se basa en la observación de patrones de consumo, donde cada *cluster* representa un perfil de comportamiento específico. Los 5 grupos permiten identificar variaciones significativas en el consumo, destacando categorías como consumo bajo, medio y alto, así como comportamientos asociados los días entre semana y fines de semana.

El gráfico resultante reportado en la Figura 17, revela la distribución de los datos en cinco clusters distintos. Los puntos de datos se agrupan en función de su proximidad a los centros de los clusters, representados por símbolos negros en el gráfico.

Se observa que algunos puntos quedan alejados de los centros de los clusters, lo que indica que estos podrían ser datos atípicos o anomalías. Estos puntos, que no se ajustan bien a los patrones de los clusters principales, podrían introducir ruido en el análisis y en el modelo de predicción. La identificación y manejo adecuado de estos datos atípicos es crucial para mejorar la precisión del modelo y reducir su susceptibilidad a influencias no representativas.

Figura 17.

Gráfico de datos atípicos, implementación del algoritmo K-Means.



Nota. Con esta grafica se quiere validar que datos en la data completa, son datos atípicos implementando el algoritmo K-Means.

5.2.3 Preprocesamiento de los Datos

La fase de preprocesamiento de datos es un paso fundamental en la preparación de datos para el desarrollo de modelos predictivos. Durante esta etapa, se realizaron varias operaciones críticas para mejorar la calidad del conjunto de datos y optimizar su utilidad para el entrenamiento y evaluación del modelo.

5.2.3.1 Eliminación de datos nulos y registros muy antiguos

Se llevó a cabo la eliminación de datos nulos, ya que los valores faltantes pueden introducir sesgos y errores significativos en el análisis. Además, se excluyó el conjunto de datos correspondiente a los años 2010 a 2014, evidenciado en la Figura 18. Esta decisión se basó en la observación de que el pronóstico de la demanda eléctrica ha mostrado una tendencia creciente a lo largo del tiempo. Los datos tan antiguos podrían no reflejar las condiciones actuales y podrían afectar negativamente la precisión del modelo. En particular, su exclusión podría resultar en un rendimiento óptimo del modelo de red neuronal LSTM, dado que las dinámicas actuales del consumo podrían no estar adecuadamente representadas por estos datos históricos. Por lo tanto, se optó por centrarse en datos más recientes para mejorar la capacidad predictiva y la relevancia del modelo.

Figura 18.

Código de eliminación de datos nulos y eliminación de data en el rango 2010 a 2014.

```
df=dataFrame
print(dataFrame.shape)
df.head()
#Eliminacion de las filas que estaban vacias
df = df.iloc[:4812]
df.head(len(df))
print(df.shape)
#Eliminar rango de fechas del registro del año 2010 a 2014
df_cont = df
# Convierte la columna de fecha a formato de fecha
df_cont['FECHA'] = pd.to_datetime(df_cont['FECHA'])
# Especifica las fechas de inicio y fin del rango a eliminar
fecha_inicio = pd.to_datetime('2010-08-02')
fecha_fin = pd.to_datetime('2015-01-01')
# Filtra el dataframe para eliminar el rango de fechas especificado
df_filtrado = df_cont[~((df['FECHA'] >= fecha_inicio) &
(df_cont['FECHA'] <= fecha_fin))]
# Imprime el dataframe resultante
print(df_filtrado.shape)
df_filtrado.head()
```

Nota. En la figura se muestra la eliminación de la data que no se va a implementar para el desarrollo del modelo predictivo.

Adicionalmente, se añadió una columna al conjunto de datos históricos denominada "Evento", como se muestra en la Figura 19. Esta columna de tipo categórico y formato de texto, captura las variables sociales relevantes, como la época de la pandemia y otros eventos que pueden causar consumos atípicos. Los valores de esta columna incluyen "Ninguno", que indica la ausencia de eventos relevantes, y "Pandemia", que se asigna a las fechas comprendidas entre el 6 de marzo de 2020 y el 1 de julio de 2022, período durante el cual se observó un comportamiento atípico en el consumo de energía debido a la pandemia de COVID-19. Al incorporar esta columna, se facilita el análisis de cómo estos eventos impactan las tendencias de consumo energético.

Figura 19.

Añade columna "Evento" adiciona variable social relevante.

```
df_filtrado.insert(3,"Evento","Ninguno",allow_duplicates=False)
## PANDEMIA
temporal="Pandemia"
df_filtrado.loc["2020-03-06":"2022-07-01","Evento"]=temporal
print(df_filtrado.loc["2020-03-06":"2022-07-01"]["Evento"])
```

Nota. En la figura se evidencia el código para añadir una nueva columna "Evento" a la estructura de la data.

5.2.3.2 Eliminación de columnas

Se eliminaron las columnas **'DEMANDA METRO'**, **'DEMANDA NORTE'**, **'DEMANDA SAN GIL'**, **'DEMANDA BARBOSA'**, **'TEMP METRO'**, **'TEMP BARRANCA'**, **'TEMP SAN GIL'** y **'TEMP BARBOSA'** del conjunto de datos, como se evidencia en la Figura 20, dado que no se utilizaron en el proceso de predicción, esta decisión se tomó porque la ESSA clasifica estos datos según factores de distribución y, para el pronóstico, se solicitó no considerar dichos valores. Esta reducción se realizó para simplificar el conjunto de datos y concentrarse únicamente en las variables relevantes para el modelo.

Figura 20.

Código eliminación de columnas que no son relevantes en el conjunto de los datos.

```
# ELiinacion de columnas para la clasificacion
drop_colums = ['DEMANDA METRO', 'DEMANDA NORTE', 'DEMANDA SAN GIL',
'DEMANDA BARBOSA', 'TEMP METRO', 'TEMP BARRANCA', 'TEMP SAN GIL',
'TEMP BARBOSA']
# print(drop_colums)
df2.drop(drop_colums, axis=1 , inplace= True)
df2.head()
```

Nota. En la figura se evidencia el código con el cual se realiza la eliminación de las columnas con data no relevante para el desarrollo del modelo predictivo.

5.2.3.3 Transformación de los datos

La transformación de datos cualitativos en valores numéricos es una etapa crucial en el preprocesamiento de datos, especialmente cuando se trabaja con modelos de aprendizaje automático que requieren entradas en formato numérico. Este proceso permite que las variables categóricas sean interpretadas y procesadas de manera efectiva por los algoritmos de modelado.

En esta subfase, se crearon diccionarios para convertir las variables cualitativas del conjunto de datos en valores numéricos, como se evidencia en la Figura 21. Los datos cualitativos, tales como, los días, tipos de días y evento fueron asignados a identificadores numéricos específicos mediante un mapeo sistemático. Este enfoque facilita la incorporación de variables categóricas en el análisis predictivo y asegura que la información cualitativa sea manejada adecuadamente por el modelo.

El uso de diccionarios para esta transformación no solo estandariza los datos, sino que también optimiza el procesamiento y la integración de estas características en el modelo

predictivo. Este paso es esencial para garantizar que todas las variables, tanto cualitativas como cuantitativas, sean representadas de manera uniforme y eficaz en el proceso de entrenamiento del modelo, ver Figura 22.

Figura 21.

Diccionarios para convertir las variables cualitativas del conjunto de datos en valores numéricos.

```
from operator import index
fechas= data_f.index.to_numpy()
# Crear un DataFrame con las fechas
df = pd.DataFrame({'fecha': fechas})
# Codificación de fechas
df['dia_del_anio'] = df['fecha'].dt.dayofyear
df['mes'] = df['fecha'].dt.month
df['año']= df['fecha'].dt.year
df['dia_de_la_semana'] = df['fecha'].dt.dayofweek
# Obtener el primer año del DataFrame
primer_año = df['año'].min()
# Etiquetar los años del 1 al 8 (considerando 2015 como año 1)
df['año'] = df['año'] - primer_año + 1

df=df.iloc[:,1:4]
#print(df)
data_=df.to_numpy()
data_1=data_f.to_numpy()
#print(data_1)
## hacer etiquetas
diccionario_dias = {
    'lunes': 1,
    'martes': 2,
    'miércoles': 3,
    'jueves': 4,
    'viernes': 5,
    'sábado': 6,
    'domingo': 7
}
```

Nota. En esta figura se evidencia el código mediante el cual se implementó el uso de los diccionarios para convertir las variables cualitativas en valores numéricos.

Figura 22.

Gráfico de data de entrada transformada.

	0	1	2	3	4	5	6	7	8	9	...	46	47	48	49	50	51	52	53	54	55
0	10	1	1	6	6	0	267.68	255.31	244.34	237.26	...	27.8	27.8	26.5	26.3	25.3	24.6	23.2	22.5	21.9	20.4
1	13	1	1	2	2	0	253.86	243.19	236.54	232.16	...	27.0	27.4	26.8	26.7	25.8	24.7	24.2	24.0	23.7	22.9
2	14	1	1	3	3	0	269.41	256.44	246.96	241.22	...	25.9	26.4	26.4	26.2	24.8	24.4	24.1	23.0	22.6	22.4
3	15	1	1	4	4	0	278.37	265.53	254.61	248.69	...	26.6	26.6	26.7	26.3	25.7	24.9	24.2	23.7	23.0	22.1
4	16	1	1	5	5	0	268.48	260.01	252.19	246.01	...	26.1	26.2	25.3	25.0	24.5	24.2	24.1	23.7	22.7	21.9

Nota. En esta figura se evidencia la estructura final de la data de entrada para el modelo predictivo.

5.3 Diseño, Arquitectura y Entrenamiento del Modelo

En este capítulo, se detalla el proceso de diseño y entrenamiento de un modelo de red neuronal Long Short-Term Memory (LSTM) para la predicción de la demanda eléctrica, que corresponde a la Fase 3 y 4 de la metodología. La red neuronal LSTM es una variante de las redes neuronales recurrentes (RNN) que es particularmente eficaz en el manejo de secuencias temporales y patrones dependientes del tiempo, lo cual es crucial para el análisis de series temporales de demanda eléctrica.

Para el diseño del modelo, se comenzó por preparar los datos para su entrada en la red neuronal. El conjunto de entrada es (30,1) por cada dato histórico, que son los 24 periodos de temperatura, fecha: día, mes, año, día, tipo de día y evento, como se evidencia en la Figura 23. El conjunto de datos fue segmentado y reformateado para cumplir con los requisitos de entrada de la LSTM, los cuales son los 24 periodos de temperatura, día(fecha), mes(fecha), año(fecha), día de la semana, tipo de día, y evento que conforman un conjunto de datos X, que puede representarse como una matriz,

$$X = \begin{bmatrix} t_1^1 & \dots & t_1^{30} \\ \vdots & \ddots & \vdots \\ t_n^1 & \dots & t_n^{30} \end{bmatrix}, \qquad \text{(Manobanda Vega, 2020)}$$

Donde t_i^j representa las características de cada muestra i en el tiempo j, que incluyen:

- $t_i^{(1)}$: Día (fecha)
- $t_i^{(2)}$: Mes (fecha)
- $t_i^{(3)}$: Año (fecha)
- $t_i^{(4)}$: Día de la semana
- $t_i^{(5)}$: Tipo de día
- $t_i^{(6)}$: Evento
- $t_i^{(7)}$ a $t_i^{(30)}$: Los 24 periodos de temperatura

Se extrajo un subconjunto de datos de datos de características (x) mostrado anteriormente y etiquetas (y).

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$ (Acuerdo 1303, 2020)

Donde y_i representa la etiqueta correspondiente para la muestra/fila i, que es el valor de la demanda de energía asociado a cada uno de los 24 periodos de tiempo.

Los datos fueron posteriormente divididos en conjuntos de entrenamiento y prueba utilizando un train_test_split, como se muestra en la Tabla 4.

Figura 23

Datos de entrada y división del conjunto de datos

```
X_menor=data_final[:,32:56]
# X_menor=scaler.fit_transform(X_menor)
X= np.concatenate((data_final[:, 0:6], X_menor), axis=1)
y= data_final[:, 6:30]
# y=scaler.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, ran-
dom_state=0, train_size = .80)
X_train = X_train.reshape(-1, 30, 1)
y_train= y_train.reshape(-1, 24)
```

Nota. En esta figura se evidencia el código implementado para tomar la data de entrenamiento y validación para el modelo de Red neuronal LSTM.

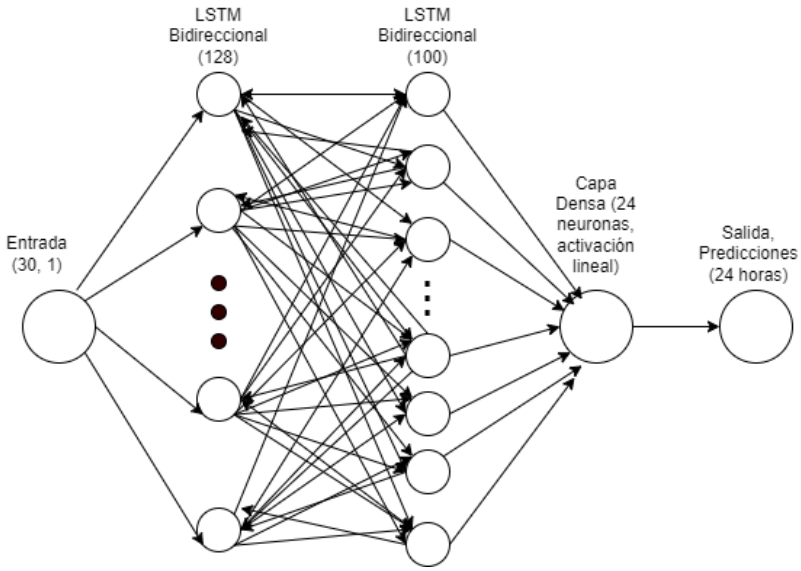
El modelo LSTM diseñado se construyó utilizando la API de Keras en TensorFlow. El diseño y arquitectura se evidencian en las Figura 24 y 25. La arquitectura del modelo incluye

capas bidireccionales LSTM, que permiten capturar dependencias temporales en ambas direcciones a lo largo de la secuencia. La estructura del modelo es la siguiente (ver Figura 24):

- **Primera capa LSTM bidireccional**: Contiene 128 unidades de memoria y return_sequences=True, lo que significa que cada celda LSTM devuelve una secuencia completa para pasarla a la siguiente capa. La cual cumple con un propósito específico en la captura de patrones temporales en los datos de entrada, asi como procesa la información temporal en ambas direcciones, mejorando la precisión de la predicción.
- **Segunda capa LSTM bidireccional**: Contiene 100 unidades de memoria y no devuelve secuencias (solo el último estado oculto), ya que se conecta a la capa de salida. Esta capa actúa como consolidación de la información procesada por la primera capa LSTM, también optimiza la información para alimentar la última capa de la red Neuronal LSTM.
- **Capa de salida densa**: Tiene 24 neuronas que corresponden a las predicciones para cada una de las 24 horas del día. La activación es linear, lo que significa que la salida es un valor numérico continuo (apropiado para una predicción de demanda).

Figura 24

Arquitectura de la red neuronal LSTM



Nota. Esta figura evidencia la arquitectura del modelo de la Red Neuronal LSTM implemnetada.

El modelo sigue una arquitectura secuencial, lo que implica que las capas se apilan una tras otra, procesando la información de manera ordenada. Los datos de entrenamiento (X_train y y_train) y de prueba (X_test y y_test) se convierten a tipo float32 para garantizar la compatibilidad, como se ilustra en la Figura 25.

Figura 25

Modelo de la red neuronal LSTM

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Bidirectional
from tensorflow.keras.metrics import MeanAbsolutePercentageError, MeanSquaredError,
Accuracy,  MeanAbsoluteError
from tensorflow.keras.metrics import Accuracy
# Definir la arquitectura de la red LSTM
model = Sequential()
model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(30, 1)))
model.add(Bidirectional(LSTM(100)))
model.add(Dense(24, activation='linear'))  # Capa de salida con 24 neuronas para las
24 horas del día
accuracy_metric= Accuracy()
mape_metric=MeanAbsolutePercentageError()
mse_metric=MeanSquaredError()
mae_metric=MeanAbsoluteError()

X_train = X_train.astype('float32')
y_train = y_train.astype('float32')
X_test = X_test.astype('float32')
y_test = y_test.astype('float32')
# Compilar el modelo
model.compile(loss='mse', optimizer='adam', metrics=[accuracy_metric,mape_metric,
mse_metric,mae_metric ])

# Entrenar el modelo con datos de validación
history = model.fit(X_train, y_train, batch_size=64, epochs=1000, valida-
tion_data=(X_test, y_test))
```

Nota. En esta figura se evidencia el código de la red neuronal LSTM usada como modelo predictivo en el proyecto.

Durante el proceso de entrenamiento, se utiliza un tamaño de lote de 64 y se realiza a lo largo de 1000 épocas. Además, se emplea un conjunto de datos de validación para evaluar el modelo y prevenir el sobreajuste (ver Figura 26). Para evaluar el rendimiento del modelo, se utilizaron varias métricas: el Error Absoluto Medio Porcentual (MAPE), el Error Cuadrático

Medio (MSE) y el Error Absoluto Medio (MAE). El modelo se optimiza con el optimizador Adam y se minimiza la función de pérdida de Error Cuadrático Medio (MSE).

Este enfoque permite que el modelo LSTM aprenda y generalice patrones temporales complejos en los datos de demanda eléctrica, mejorando así la precisión y la capacidad predictiva del sistema.

Figura 26.

Gráfico de data de para pruebas de la red neuronal.

```
X_test = np.array([
    # Día, Mes, Año, Día, de la semana (1 al 7), Tipo de día (1 al 12),
Evento, Temperaturas (24 valores)
    [296,10,9,1,8,0, 24.6,  24.7, 23.9, 23.5, 24, 23.6, 23.2, 23.4, 23.5,
24.2, 25.7, 25.6, 27.2, 27.8, 27.3, 28.8, 26.9, 27.4, 27.8, 27.8, 26.7,
25.9, 25.3, 25.1],
    [297,10,9,2,2,0, 24.6,  24.6, 24.1, 22, 22.7, 22.1, 21.8, 22.4, 20.9,
23.1, 27.1, 28.1, 30.3, 31.8, 32.4, 31.1, 29.8, 29.2, 28.9, 28.9, 28.3,
27.7, 27, 23.3],
    [298,10,9,3,3,0, 22.7,  22.5, 22.4, 23.2, 21.8, 21.8, 21.9, 21.3, 21.3,
24, 25.8, 28.4, 29.4, 30.1, 28.3, 27.4, 26.7, 26.3, 26.8, 26.4, 25.7, 25.6,
25.5, 25.3],
    [299,10,9,4,4,0, 24.6,  24.5, 24.3, 23.6, 23.5, 21.8, 21.6, 22.9, 22.2,
24.4, 25.8, 28, 28.4, 29.7, 27.8, 27, 26.1, 26.6, 26.4, 25.2, 24.5, 23.9,
23.7, 23.2],
    [300,11,9,5,5,0, 23.2,  23.4, 23.4, 23, 21.9, 21.7, 22.1, 22.2, 21.8,
24.1, 24.3, 26.3, 27, 29, 29.1, 28.6, 27.5, 26.7, 26.4, 25.7, 25.2, 24.9,
25.2, 24.6],
    [308,11,9,6,6,0, 22.6,  22.6, 21.7, 21.6, 21.7, 21.6  ,21.5,  22.7, 25,
26.9, 28.9, 30.2, 30.7, 30.1, 29.2, 27.4, 26.7, 25.9, 25.3, 25.1, 24.9
,24.6,  24.3, 23.8],
    [302,11,9,7,7,0, 23.9,  23.5, 24, 23.6, 23.2, 23.4, 23.5, 24.2, 25.7,
25.6, 27.2, 27.8, 27.3, 28.8, 26.9, 27.4, 27.8, 27.8, 26.7, 25.9, 25.3,
25.1, 24.6, 24.6]
])
# X_test = scaler.fit_transform(X_test)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Realizar predicciones
predictions = model.predict(X_test)

predictions = predictions.reshape(-1, 24)

print(predictions)
```

Nota. En el grafico se evidencia el código para testear el modelo predictivo de la red neuronal LSTM implementada.

5.4 Validación y Pruebas del Modelo

Esta etapa corresponde a la Fase 5 de la metodología y se divide en dos partes. La primera consiste en la validación del modelo utilizando un conjunto de datos de prueba, lo que permite evaluar su precisión y capacidad de generalización. La segunda parte implica un monitoreo del modelo durante semanas, donde se contrastan y verifican las predicciones realizadas con los datos de consumo real actual. Este enfoque no solo permite evaluar la efectividad del modelo, sino que también proporciona información valiosa para realizar ajustes y mejoras en futuras iteraciones del pronóstico.

5.4.1 Datos de prueba de la Red Neuronal LSTM

Para evaluar el rendimiento del modelo LSTM, se preparó un conjunto de datos de prueba (X_test) que incluye información estructurada de manera detallada, como se muestra en la Figura 26. Cada fila del conjunto de datos representa una instancia con características relevantes para la predicción de la demanda eléctrica a lo largo de las 24 horas del día. A continuación, se describe la estructura y el proceso de preparación del conjunto de datos de prueba:

Estructura de los Datos de Prueba: El conjunto X_test está compuesto por los siguientes atributos para cada día:

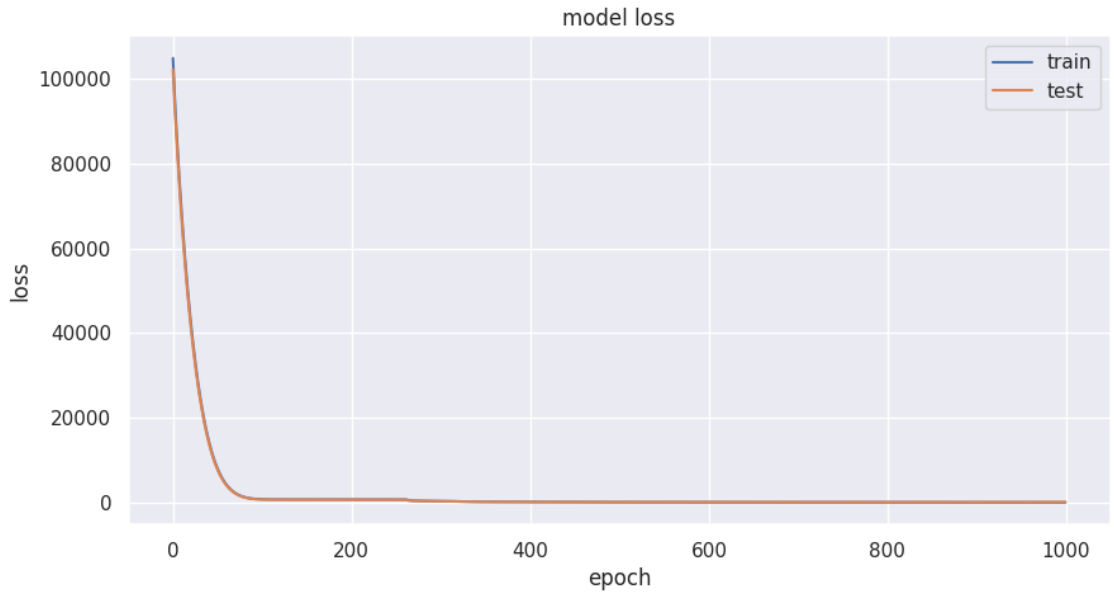
- **Fecha**: Día, Mes y Año.
- **Día de la Semana**: Representado como un valor numérico del 1 al 7.

- **Tipo de Día**: Representado con valores del 1 al 12, indicando diferentes tipos de días (por ejemplo, festivos, laborales, etc.).
- **Evento**: Un valor numérico que identifica eventos especiales, como festividades o situaciones atípicas.
- **Temperaturas Horarias**: 24 valores correspondientes a las temperaturas registradas a lo largo de las 24 horas del día.

Estos datos se presentan en un formato estructurado, con una fila por día y columnas que representan cada uno de los atributos mencionados.

Figura 27

Gráfico del rendimiento del modelo de la Red Neuronal LSTM.



La Tabla 4, resume la información clave del conjunto de datos utilizado en el proyecto, desglosando el proceso en tres etapas principales. Durante la fase de limpieza y preparación de los

datos se procesaron 1,653 registros, lo que representa el 34% del total. La mayoría de los datos, un 56%, fue empleada en la fase de entrenamiento del modelo, con un total de 2,713 registros. Finalmente, el 10% restante, correspondiente a 485 registros, fue reservado para la fase de prueba. En total, el conjunto de datos consta de 4,851 entradas, asegurando un número de valores considerable para el entrenamiento y evaluación del modelo predictivo.

Tabla 4.

Información relevante del conjunto de datos.

Proceso	***#Datos***	***%Datos***
Limpieza y preparación de los datos	*1653*	*34%*
Entrenamiento	*2713*	*56%*
Prueba	*485*	*10%*
Total	*4851*	*100%*

Nota. Esta tabla muestra la división de los datos en el pre-procesamiento y el desarrollo del modelo predictivo.

5.4.2 Monitoreo de pronóstico a partir del modelo de la red neuronal

El monitoreo de la red neuronal se llevó a cabo durante varias semanas, obteniendo resultados positivos en cuanto a la precisión de las predicciones. Como se muestra en la Tabla 5, los valores de MAPE obtenidos reflejan un comportamiento favorable del modelo, con márgenes de error reducidos en varias fechas clave.

En la tabla 5 se reportan datos de monitoreo de las predicciones durante los dias de tres semanas, se puede observar una tendencia hacia la reducción de los valores de MAPE, lo que indica una mejora en la precisión del modelo a lo largo del tiempo. En la primera semana, los valores de MAPE son variables, alcanzando un máximo de 5.77 el 9/10/2023 y un mínimo de 1.62 el 11/10/2023. Esto sugiere que el modelo estaba ajustándose al comportamiento de los datos. La segunda semana presenta una mayor estabilidad, con valores de MAPE más consistentes y generalmente más bajos, con un mínimo de 1.55 el 18/10/2023 y un máximo de 4.83 el 21/10/2023. Finalmente, en la tercera semana, se observa una tendencia más marcada hacia la disminución del error, alcanzando su punto más bajo de 0.80 el 27/10/2023.

Tabla 5.

Pronóstico de 3 semanas.

SEMANA	FECHA	MAPE
1	9/10/2023	5,776405
1	10/10/2023	4,88773788
1	11/10/2023	1,62497613
1	12/10/2023	2,17052576
1	13/10/2023	2,46296558
1	14/10/2023	4,81978395
1	15/10/2023	4,20901112
2	16/10/2023	3,51896582

2	17/10/2023	3,43901592
2	18/10/2023	1,555412
2	19/10/2023	1,62461483
2	20/10/2023	2,53714261
2	21/10/2023	4,83444679
2	22/10/2023	2,98439981
3	23/10/2023	1,63317367
3	24/10/2023	3,14629559
3	25/10/2023	2,1231373
3	26/10/2023	1,491471

Nota. Esta tabla muestra la evaluación de la red neuronal entrenada durante determinado tiempo.

5.5 Desarrollo de Prototipo de Software

En esta sección, se aborda el desarrollo del prototipo de software diseñado para la implementación y visualización de la red neuronal entrenada, que corresponde a la Fase 6 de la metodología. Utilizando la biblioteca Tkinter de Python, se realizó una interfaz gráfica de usuario (GUI) que facilita la interacción con el modelo de red neuronal y la visualización de los resultados de las predicciones. El objetivo principal es proporcionar una herramienta accesible y funcional para la gestión y análisis de datos, así como para la presentación de los resultados de manera clara y comprensible.

5.5.1 Requerimientos Funcionales y no funcionales

En el desarrollo de la interfaz gráfica realizada con Tkinter para la visualización y manipulación de datos de pronóstico de demanda de energía, se definieron claramente los requisitos funcionales y no funcionales como se muestra en la Tabla 6 y Tabla 7.

Tabla 6.

Requerimientos funcionales GUI.

REQUERIMIENTOS FUNCIONALES
I. Debe haber opciones claramente diferenciadas para seleccionar entre el pronóstico diario y el pronóstico semanal.
II. Permitir al usuario introducir las fechas para las cuales desea generar el pronóstico. Estos campos deben aceptar entradas de fecha en formatos válidos y validar la entrada del usuario.
III. Un botón para iniciar el proceso de generación de predicciones basado en las fechas introducidas
IV. El sistema debe procesar los datos y ejecutar el modelo de red neuronal LSTM para generar las predicciones.
V. Un botón para generar un reporte en formato tabla con las predicciones diarias o semanales.
VI. El reporte debe presentarse de manera clara y comprensible, mostrando la predicción para cada período solicitado.

Nota. Definición de los requerimientos funcionales para el desarrollo de la interfaz gráfica

Tabla 7.

Requerimientos no funcionales GUI.

REQUERIMIENTOS NO FUNCIONALES
I. La generación de predicciones y la creación de reportes deben completarse en un tiempo razonable para no afectar la experiencia del usuario. Idealmente, el tiempo de respuesta debería ser de unos pocos segundos. II. El sistema debe ser capaz de manejar errores de entrada de forma robusta y proporcionar mensajes de error claros al usuario. III. La interfaz debe ser fácil de usar y navegar, con etiquetas claras y botones de acción evidentes.

Nota. Definición de los requerimientos funcionales para el desarrollo de la interfaz gráfica

5.5.2 Diseño de la Interfaz de Usuario

El diseño de la interfaz se centra en ofrecer una experiencia de usuario intuitiva y eficiente. La GUI desarrollada incluye las siguientes secciones:

Sección de Carga de Datos: Permite al usuario cargar archivos de datos históricos necesarios para realizar las predicciones. Se incluye la opción de seleccionar archivos en formatos como .csv o .xlsx.

Sección de Predicción: Permite ejecutar predicciones en tiempo real basadas en los datos cargados y los parámetros configurados. Los resultados de las predicciones se visualizan en gráficos y tablas.

Visualización de Resultados: Incluye gráficos interactivos que muestran las predicciones de demanda eléctrica a lo largo del tiempo, comparados con los datos reales. También se generan reportes en formato Excel que se pueden descargar.

Sección de Información y Ayuda: Proporciona información sobre el funcionamiento de la aplicación y ayuda para resolver problemas comunes.

5.5.2.1 Mockups

En el proceso de desarrollo de la interfaz gráfica, se crearon varios **mockups** que sirvieron como representaciones visuales preliminares del diseño y la funcionalidad esperada. Estos mockups desempeñaron un papel crucial en la planificación y conceptualización del desarrollo, proporcionando una visión clara de cómo se organizaba y se visualizaba la interfaz, algunos de estos mockups se reportan en la Figura 28.

Figura 28

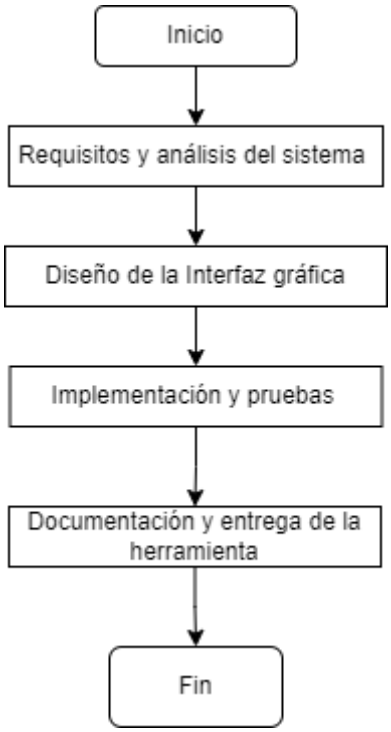
Gráfico de planteamiento del funcionamiento de la interfaz – Pronóstico semanal- pronóstico diario- Reentrenamiento de la red Neuronal.

5.5.3 Implementación con Tkinter

La implementación de la interfaz de usuario se realizó utilizando la biblioteca Tkinter de Python, que permite la creación de aplicaciones gráficas con un enfoque orientado a eventos. A continuación, se detalla el proceso de desarrollo, ver Figura 29.

Figura 29

Gráfico de desarrollo de la interfaz gráfica.



Nota. Este diagrama evidencia las actividades para el desarrollo de la interfaz gráfica.

En la Figura 29, se evidencia el flujo de desarrollo donde se tiene la definición, los requisitos y funcionalidades de la herramienta, identificando las necesidades del usuario, como la capacidad de cargar datos, realizar pronósticos y visualizar resultados mediante gráficos. Se procede con el desarrollo de la interfaz gráfica, creando la estructura básica con Tkinter y organizando los elementos visuales, como botones, cuadros de texto y áreas de visualización. En la fase de implementación, se integran las funcionalidades deseadas como la visualización de las predicciones. Posteriormente, se realizan pruebas de la usabilidad de la interfaz. Finalmente, se elabora la entrega la herramienta, donde visualizan los resultados del modelo.

Como parte del desarrollo de la interfaz gráfica se realizan los siguientes pasos:

Configuración Inicial: Se configuró la ventana principal de la aplicación, estableciendo el título, el tamaño y el diseño general utilizando el gestor de geometría de Tkinter.

Desarrollo de los Widgets: Se crearon y configuraron los diferentes widgets necesarios, como botones, cuadros de texto. Cada widget se asoció con funciones específicas para manejar eventos y realizar operaciones, como cargar datos o ejecutar predicciones.

Funcionalidades Clave:

- Opción para cargar nuevos datos y generar predicciones en tiempo real.
- Visualización de los resultados en gráficos interactivos y tablas exportables.
- Botones para guardar los pronósticos generados y emitir reportes en formatos accesibles (PDF, CSV).

Integración con el Modelo: Se integró el modelo de red neuronal LSTM entrenado con la interfaz gráfica. Las funciones de predicción y visualización se vincularon a eventos generados por la interacción del usuario con la GUI.

Visualización de Datos: Para la visualización de resultados, se utilizaron bibliotecas adicionales como Matplotlib para generar gráficos que se incrustan en la aplicación Tkinter. Los resultados de las predicciones se muestran en gráficos de líneas y barras, facilitando la interpretación de los datos.

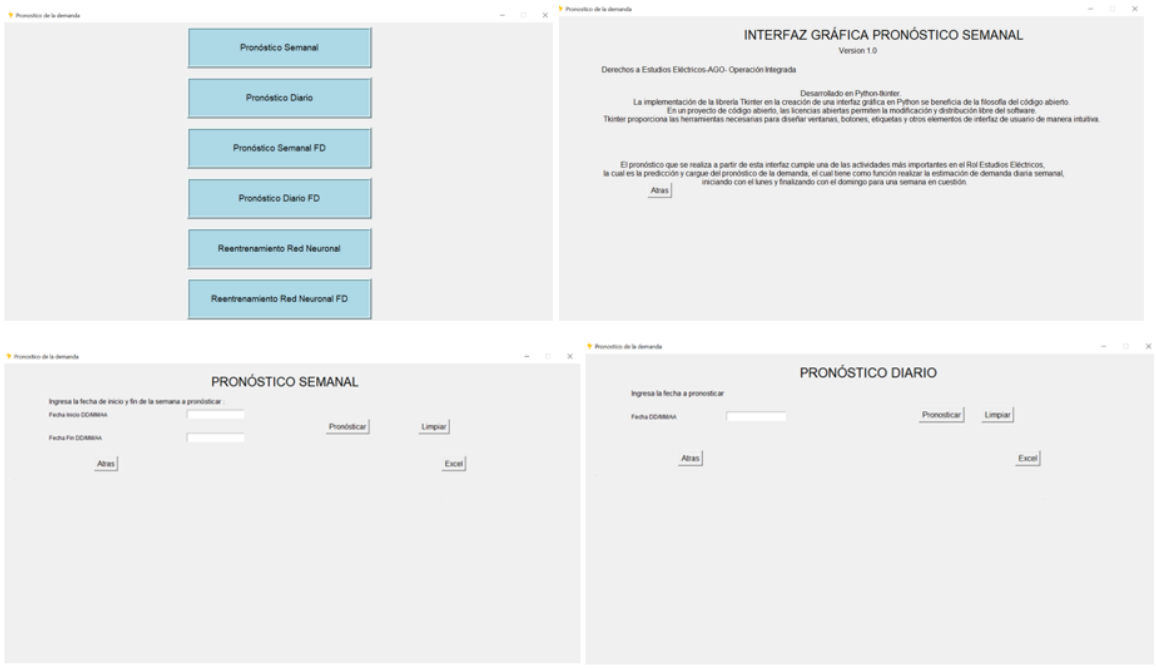
Generación de Reportes: Se implementaron funciones para exportar los resultados de las predicciones a archivos Excel. Esto permite a los usuarios realizar un análisis más detallado de los datos y compartir los resultados de manera eficiente.

Finalmente la interfaz con las funcionales descritas anteriormente se evidencian en la Figura 30, donde la estructura de este menú está en consonancia con la configuración inicial y la navegación definida durante el desarrollo (se dejaron 2 cajas para próxima implementación del pronóstico semanal por Factores de distribución).

Las ventanas correspondientes al pronóstico semanal y diario permiten ingresar datos específicos, como las fechas a pronosticar, facilitando así la interacción entre el usuario y el sistema. Estas pantallas son representativas de la etapa de desarrollo de widgets, donde los botones de pronóstico y exportación a Excel están configurados para cumplir con las funcionalidades clave mencionadas, como la generación de reportes y visualización de resultados (ver Figura 31).

Figura 30

Gráfico de la portada, versión de la interfaz creada, pronóstico semanal, pronóstico diario.Fuente: Autor



5.5.4 Pruebas y Validación

Se llevaron a cabo pruebas para asegurar que el prototipo funcione correctamente en diferentes escenarios y con diversos conjuntos de datos, ver Figura 31. Las pruebas incluyeron:

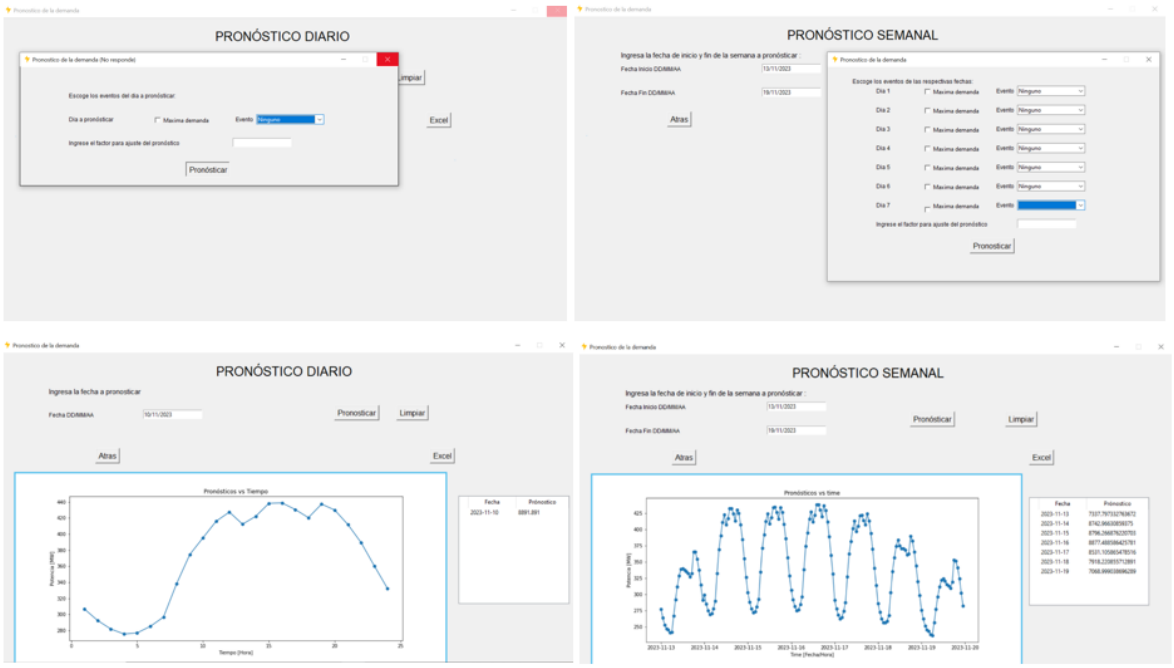
Pruebas de Carga de Datos: Verificación de la capacidad del prototipo para manejar diferentes formatos y tamaños de archivos.

Pruebas de Interacción: Evaluación de la respuesta de la interfaz a las acciones del usuario, como ajustes de parámetros y ejecución de predicciones.

Pruebas de Visualización: Validación de la precisión y claridad de los gráficos y reportes generados.

Figura 31

Gráfico de resultados, versión de la interfaz creada, pronóstico semanal, pronóstico diario.



La aplicación se ha entregado como un archivo de Python que los usuarios deben ejecutar en su entorno local para acceder a la interfaz gráfica. Esto se debe a restricciones de permisos impuestas por la entidad, que impidieron la creación de un ejecutable independiente. La aplicación requiere acceso a Internet, ya que consume una API de temperaturas para obtener el pronóstico necesario. Una vez que el usuario ejecuta el archivo en Python, se inicia la interfaz gráfica, que invoca el modelo de pronóstico desde otro archivo de Python, asegurando una integración efectiva entre ambos.

6 Conclusiones

El desarrollo del prototipo software para la predicción del consumo de energía utilizando redes neuronales LSTM, llevado a cabo bajo la modalidad de prácticas empresariales, ha demostrado ser una herramienta efectiva en la estimación a corto plazo de consumo, proporcionando resultados precisos que pueden ser aplicados en la planificación y gestión del consumo energético. La incorporación de datos históricos, incluyendo variables como la temperatura, ha permitido capturar patrones complejos en el comportamiento de la demanda eléctrica, lo que refuerza la importancia de un preprocesamiento adecuado y la selección de variables relevantes.

La experiencia en el entorno empresarial ha sido particularmente enriquecedora, permitiendo una integración efectiva entre el conocimiento teórico adquirido durante la formación académica y su aplicación práctica en un contexto real. Esto ha facilitado enfrentar desafíos concretos y adoptar soluciones tecnológicas que responden a necesidades específicas del sector energético, destacando cómo la innovación, como la implementación de redes neuronales y el desarrollo de software, puede optimizar procesos clave.

La interfaz gráfica facilita la interacción con el modelo, la gestión de datos y la presentación de resultados, contribuyendo a una mejor comprensión y análisis de las predicciones de demanda eléctrica.

El uso de Tkinter para la construcción de la interfaz gráfica ha resultado en una solución funcional y accesible para el usuario final, facilitando la interacción con el sistema y permitiendo la visualización clara de las predicciones. La posibilidad de generar reportes y de seleccionar intervalos de tiempo específicos proporciona una flexibilidad que puede adaptarse a diversas necesidades en la toma de decisiones.

En resumen, este proyecto no solo ha permitido alcanzar los objetivos propuestos en cuanto al desarrollo del software de predicción de energía, sino que también ha contribuido significativamente al crecimiento profesional, brindando una perspectiva más amplia y realista del impacto que las tecnologías de la información pueden tener en el sector energético. Además, se ha evidenciado la viabilidad de aplicar tecnologías como el aprendizaje profundo en la predicción del consumo energético, subrayando la importancia de seguir mejorando estos modelos con datos adicionales y técnicas más avanzadas.

Referencias

Acuerdo 1303 por el cual se actualizan los procedimientos para la gestión integral de la demanda. (2020).

Alharthi, A., & Jebelli, M. (2021). Implementation of deep learning models for electricity demand forecasting using Python and Tkinter. *Journal of Computational and Applied Mathematics, 394*, 113572.

Budde, R., Kautz, K., Kuhlenkamp, K., & Zullighoven, H. (1992). *Prototyping: An approach to evolutionary system development*. Springer-Verlag.

Brooks, F. P. (1987). *No Silver Bullet: Essence and Accidents of Software Engineering*. ACM SIGSOFT Software Engineering Notes, 10(1), 10-19.

Descubre TensorFlow: La revolución en aprendizaje automático. (2023). Aprender Big Data. https://aprenderbigdata.com/tensorflow/

Grayson, J. (2000). *Python and Tkinter programming*. Manning Publications.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.

Hong, T., & Fan, S. (2016). *Probabilistic electric load forecasting*: A tutorial review. *International Journal of Forecasting, 32*(3).

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: Concepts and techniques*. Elsevier.

Keras: Todo sobre la API de deep learning. (s.f.). DataScientest. https://datascientest.com/es/keras-la-api-de-deep-learning

López, F., & Martínez, J. (2019). *Optimización de modelos de predicción de demanda eléctrica en sistemas de distribución. Revista de Ingeniería Eléctrica y Electrónica, 21*(2), 45-62.

La guía definitiva del paquete NumPy para computación científica en Python. (s.f.). *freeCodeCamp*. https://www.freecodecamp.org/espanol/news/la-guia-definitiva-del-paquete-numpy-para-computacion-cientifica-en-python

Manobanda Vega, A. F. (2020). *Predicción de la demanda de energía eléctrica en la producción de petróleo de los campos de Petroamazonas EP utilizando redes neuronales artificiales*.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

Olah, C. (2015). *Understanding LSTM Networks*. Colah's Blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Open-Meteo. (n.d.). *Open-Meteo*. Retrieved August 12, 2024, from https://open-meteo.com

Pérez, J. (2021). *Desarrollo de un modelo predictivo para la demanda de energía eléctrica utilizando redes neuronales* (Trabajo de grado). Universidad Técnica del Norte. https://repositorio.utn.edu.ec/bitstream/123456789/10658/2/04%20IND%20271%20TRABAJO%20GRADO.pdf

Paladino, S. (2022). *The role of advanced forecasting techniques in the energy sector. Energy Research & Social Science*.

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.

¿Qué es matplotlib y cómo funciona? (s.f.). *KeepCoding*. https://keepcoding.io/blog/que-es-matplotlib-y-como-funciona/

¿Qué es una API? (s.f.). *Amazon Web Services*. https://aws.amazon.com/es/what-is/api/#:~:text=API%20significa%20%E2%80%9Cinterfaz%20de%20programaci%C3%B3n,de%20servicio%20entre%20dos%20aplicaciones.

Rudd, J., Stern, K., & Isensee, S. (1996). *Low vs. high-fidelity prototyping debate*. *Interactions, 3*(1), 76-85.

Reinoso, F. (2022). *Predicción del consumo eléctrico mediante redes neuronales LSTM*. Repositorio Digital UTEQ.

Royce, W. W. (1970). *Managing the Development of Large Software Systems*. Proceedings of IEEE WESCON, 1-9.

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.

Weron, R. (2014). *Electricity price forecasting: A review of the state-of-the-art with a look into the future*. *International Journal of Forecasting, 30*(4), 1030-1081. https://doi.org/10.1016/j.ijforecast.2014.08.007

Zhang, G., Qi, M., & Wang, H. (2020). *Integration of renewable energy and electricity demand forecasting. Energy Reports, 6*, 493-505.